INTRODUKTION TILL PROGRAMMERING

BASIC FÖR NYBÖRJARE



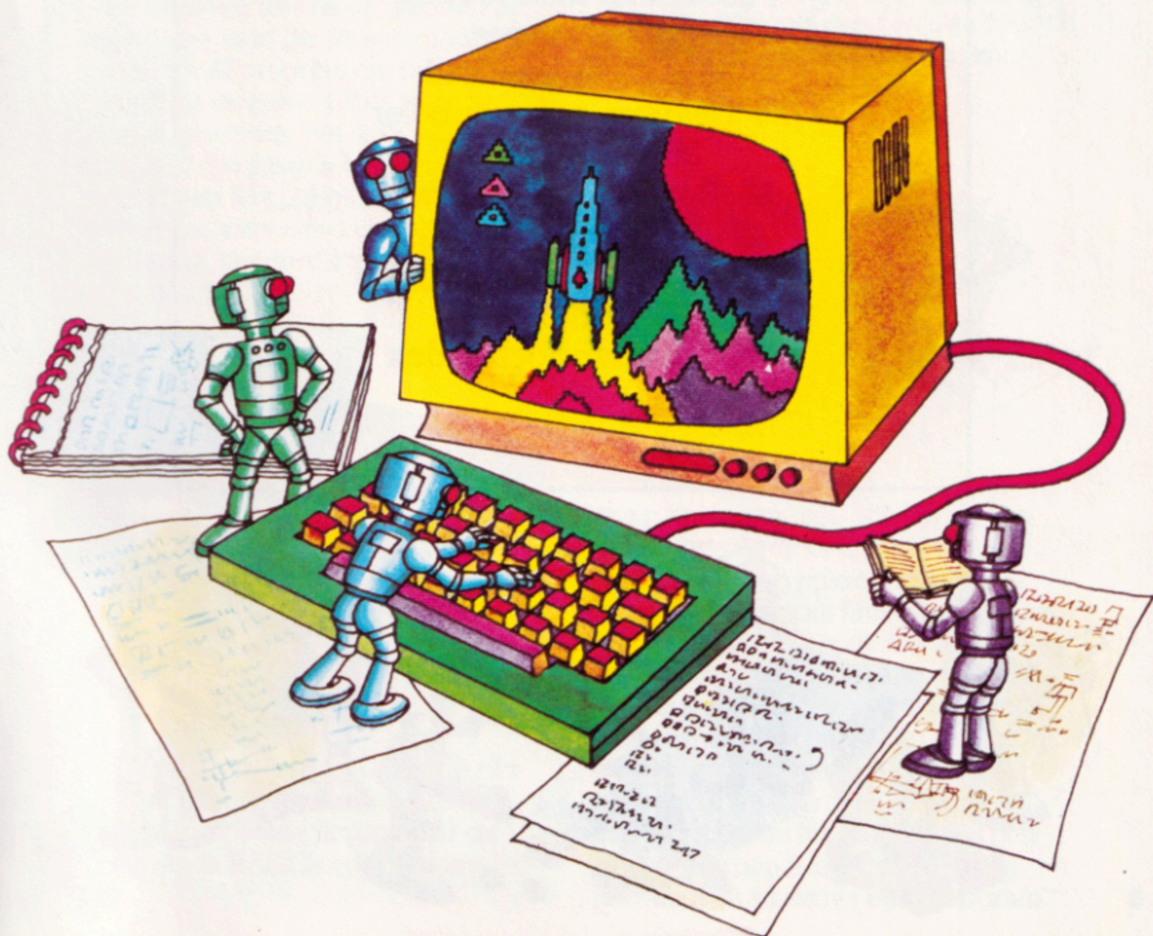
BROMBERGS

DATOR BEHÖVS EJ

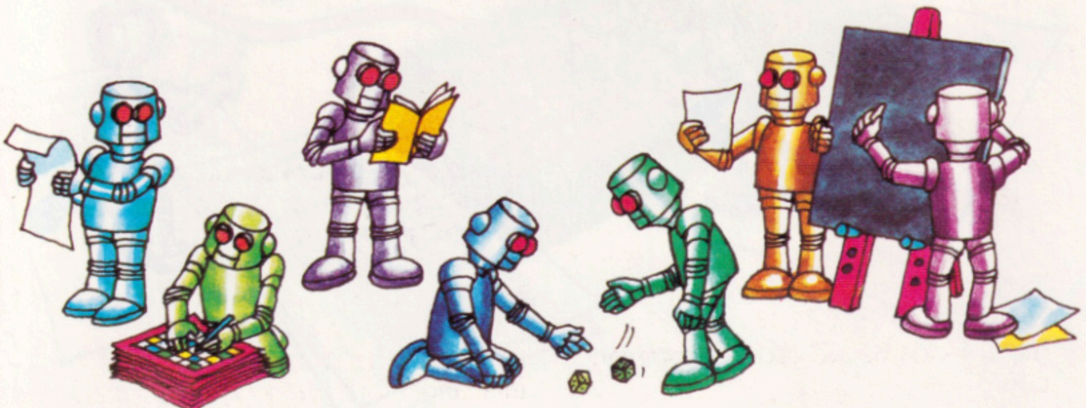
INTRODUKTION TILL PROGRAMMERING

BASIC FÖR NYBÖRJARE

Redaktion:
Tad Gruber och Gull-Mari Lenderud



Innehåll



Introduktion till programmering

Käre läsare,
Från Magnus Beckman har vi fått följande råd om justeringar av programmen i boken:

På sidan 16 ska den rosa rutans översta rad vara:

JAG ÄR UPPDELAD (avståndet mellan andra och tredje ordet ska ökas)

I kodprogrammet på sid 33 ska rad 90 vara:

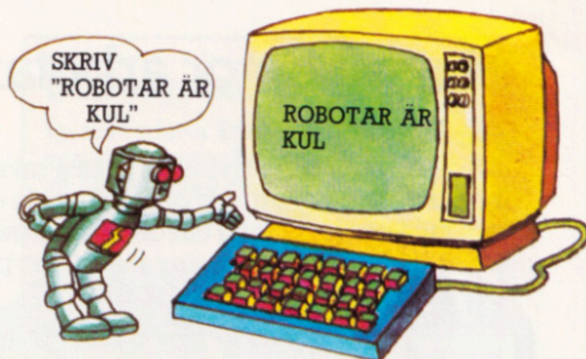
```
90 LET C$=C$+MID$(M$,I,1)
```

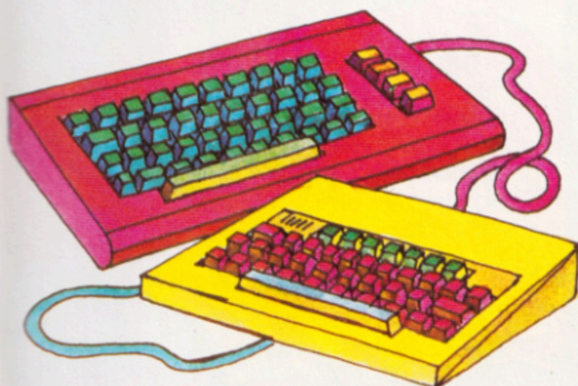
På sidan 46 är näst sista raden i rutan med BASIC-ord lite oklar. Det bör istället stå:

LEFT$(A$,4) betyder tag fyra tecken från vänster i A$.

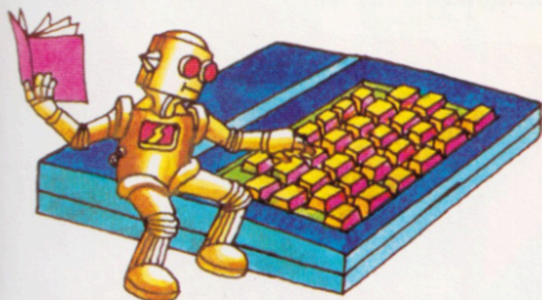
Bokens innehåll

Det här är en bok för dig som vill lära dig programmera datorer i BASIC. BASIC är ett programmeringsspråk som används på de flesta hemdatorer. En hemdator är en mikrodator.





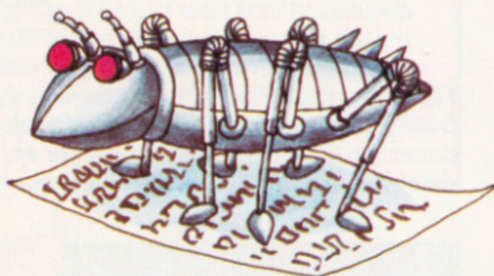
Du behöver inte ha en dator för att läsa boken, men det är naturligtvis lättare att förstå program om du kan pröva dem på en dator. Olika typer av datorer använder litet olika versioner av BASIC. De flesta BASIC-ord som nämns i den här boken kan användas för flertalet mikrodatorer och de få som inte är standard kommer att tydligt markeras.



I början av boken får du några riktlinjer för programmering av datorer. Med hjälp av enkla program lär du dig använda BASIC-ord och kommandon.

För att du ska få öva dig i att skriva program innehåller boken en del problem som måste lösas och förslag till program. Du lär dig också genom att ändra i program som finns i boken och se vad som händer. Svar till övningsuppgifterna finns på sidan 44 och 45.

I slutet av boken finns en förteckning över BASIC- och andra datorord tillsammans med en kort förklaring. Du får några riktlinjer som hjälper dig att skriva program samt en lista över vanligt förekommande fel som kan



stoppa programmet, och hur du kan känna igen dem.

Om du har en mikrodator kan du pröva program som finns i boken. Det kan hända att du upptäcker att en del regler i den här boken inte gäller för din dator. I handboken står de speciella regler som gäller din dator. Det bästa sättet att lära sig BASIC är att testa program från olika böcker och tidningar, sedan ändra dem litet och se vad som händer. Därifrån är det inte långt till att skriva egna program.

Hur en dator arbetar

Hur en dator arbetar
Med en dator kan du göra flera olika saker. Du kan rita bilder eller diagram, skriva poesi, spela eller leka med ord och tal. Datorn tar hand om de data du ger den, bearbetar dem enligt dina instruktioner (ditt program) och presenterar resultat på en bildskärm eller någon annanstans som du har bestämt.

DELA 999 MED 27

SKRIV MITT NAMN 80 GÅNGER

RITA 1000,100

FÖR IN "RAKET KOMMANDO"

KÖR 36876,15

För att en dator skall göra det du vill måste du ge den mycket exakta instruktioner. En lista med instruktioner för en dator kallas program och den information datorn får för bearbetning kallas för data. Program måste skrivas i ett språk, t ex BASIC, som datorn förstår och måste följa språkets alla regler.

Mikrodatorer



De flesta mikrodatorer är inbyggda i ett tangentbord som kopplas till en vanlig TV. Instruktioner och data matas in i mikrodatorn från tangentbordet. Allt som skrivs på tangentbordet visas också på skärmen.

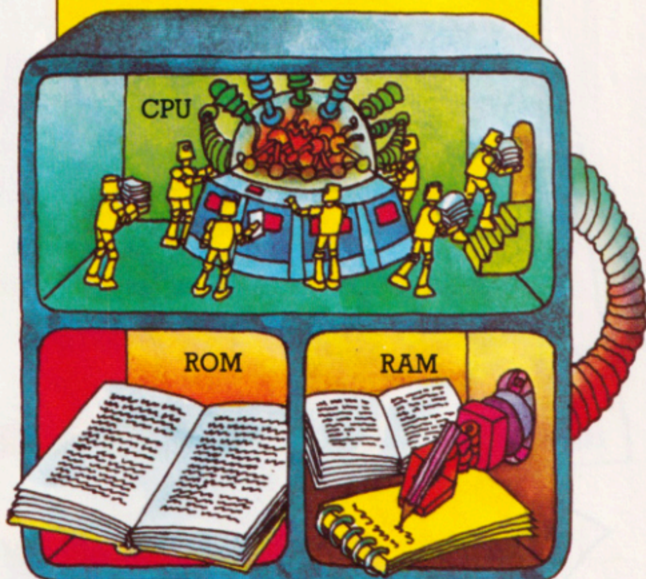
En del mikrodatorer har en liten skärm inbyggd, ungefär som en fickkalkylator. Några andra använder en speciell bildskärm kallad monitor. En monitor fungerar som en TV men utan den del som tar emot TV-signaler.

Tangentbordet ser ut som på en vanlig skrivmaskin, men har några extra tangenter. På vissa mikrodatorer kan ett helt BASIC-ord matas in med hjälp av en enda tangent.



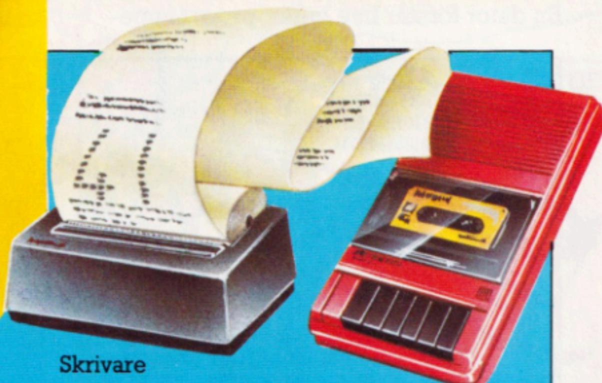
Inuti en mikrodator

En mikrodator är uppbyggd av två huvuddelar: centralenheten (CPU) där all bearbetning sker, och minnet där program och data finns.



En dator har två olika slags minnen. Ett, som kallas ROM, innehåller program som styr datorns arbete. Det andra, RAM, är tomt från början och där skrivs dina program och data in. När du stänger av din mikrodator försvinner allt som finns i RAM, medan innehållet i ROM finns kvar.



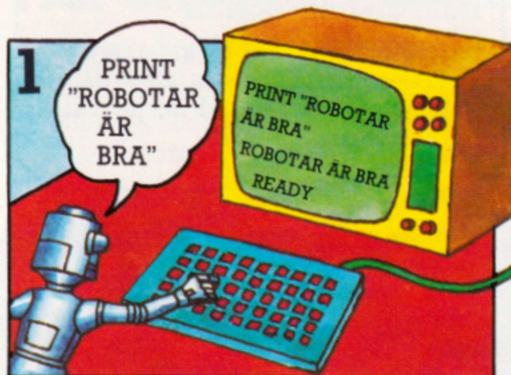


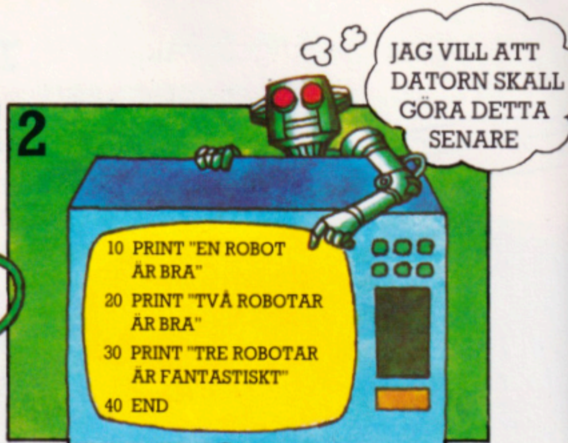
Skrivare

Information från en mikrodator visas oftast på en bildskärm, men det går också att få den utskriven på papper med hjälp av en skrivare.

Ett sätt att lagra information för att sedan vid behov ladda den tillbaka till mikrodatorn är att använda en kassettbandspelare. Både program och data kan lagras på en kassett.

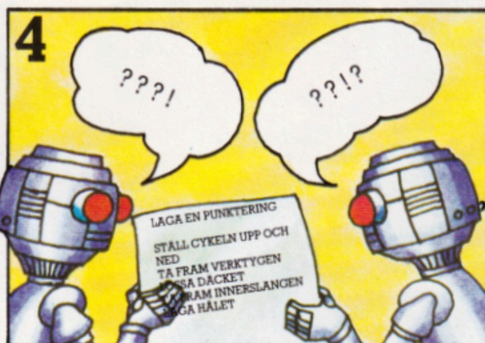
Instruera en dator





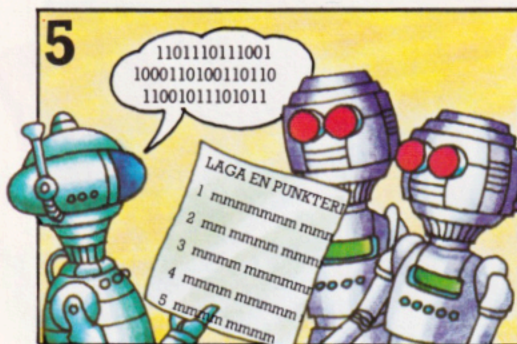
För att en dator skall göra någonting måste du mata in instruktioner som den förstår. En instruktion kan vara ett kommando som utförs omgående eller också ett flertal instruktioner som lagras i minnet och kan användas senare.

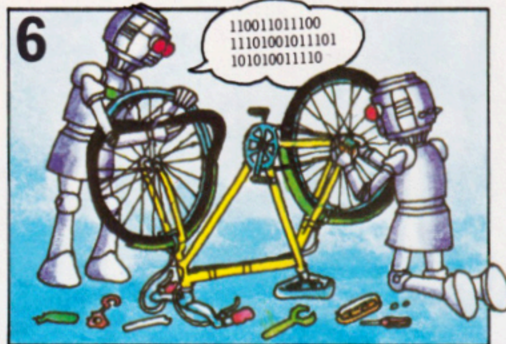




Du måste vara mycket noga när du skriver in instruktioner. En dator kommer att försöka utföra en instruktion även om den är felaktig.

En dator förstår inte heller programmeringsspråken direkt. Allt arbete i en dator sker med hjälp av svaga elektriska impulser. Dina instruktioner måste översättas till dessa impulser av ett speciellt program inuti datorn som heter tolk, eller interpret.





En dator förstår inte instruktioner skrivna på svenska, så du måste skriva dem i något av de speciella programmeringsspråk som datorer förstår. Några av dem kan du se på sidan 7.

I en dator kan program och data sägas vara en serie pulser. Maskinkod kan skrivas direkt med hjälp av "ettor" (1 = puls) och "nollor" (0 = ingen puls).

Programmeringsspråk

Du skulle kunna skriva program direkt i maskinkod, men det är mycket svårt. Istället används speciella programmeringsspråk.

Det finns många olika språk. En del av dem har utvecklats för speciella ändamål. BASIC (Beginners All-purpose Symbolic Instruction Code) är ett av de mest använda programmeringsspråken. Det används av många fler än nybörjare. Här nedan får du exempel på tre olika programmeringsspråk.

```
NAMNPROGRAM
10 PRINT "VAD HETER DU"
20 INPUT N$
30 IF N$ = "KALLE"
   THEN PRINT "HEJ"
40 IF N$ = "ANDERS"
   THEN PRINT "STICK"
50 END
```

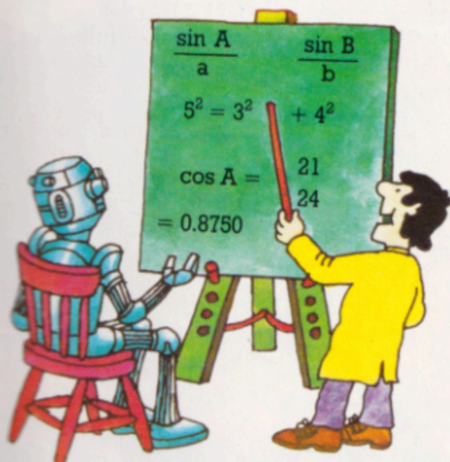
```
EKONOMIPROGRAM
FINANSIELLT FÖRFARANDE;
MÅNATLIGT: TOTALSUMMERA
BÖRJA (+KALKYLERA+)
PER MÅNAD: = 1 TO
TOTALMÅNADER DO
SUMMASKULD:=
(SUMMASKULD–ÅTERBETALA)*
(1+MÅNADSRÄNTEBELOPP)
END; (EKONOMI)
```

```
GEOGRAFISKT PROGRAM
T: VAD HETER FINLANDS HUVUDSTAD
+INPUT A: $A$
T(LEN (A$) = 0):FÖRSÖK, GISSA
JC:@ A
M: KÖPEN!STOCK!OSLO!GÖTE!LAP
TY: NÄRA MEN INTE RÄTT – FÖRSÖK IGEN
JY: @ A
M: HELSINKI! HELSINGF
TY: RÄTT DET ÄR HELSINKI: HELSINGFORS
```

Det här är ett kort program i BASIC. Rad 10 säger till datorn att på bildskärmen skriva ut frågan "Vad heter du?". Ditt svar lagras sedan i minnet och om du svarade med t ex Anders eller Kalle så får du ett meddelande.

Detta program har skrivits i Pascal som uppkallats efter en känd fransk matematiker. Det är en del av ett större program som gör ekonomiska beräkningar.

Det här språket kallas Pilot och används för undervisning. Med hjälp av Pilot kan datorn känna igen svaren även om dessa inte är helt korrekta.





11. Bb3, Ne5
12. 0-0-0, Nc4
13. Bxc4, Rxc4
14. h5, Nxh5



I början kanske du tycker att programmeringsspråk är konstiga och svåra, men så är det ju också med andra språk innan du har lärt dig dem – exempelvis finska som visas i bilden till höger. Det finns många andra områden där speciella språk används. I matematik t ex presenteras ideer och formler som skulle kräva långa förklaringar med hjälp av speciell notation. Samma gäller musik och schack.

* Minus femton, gissar jag.

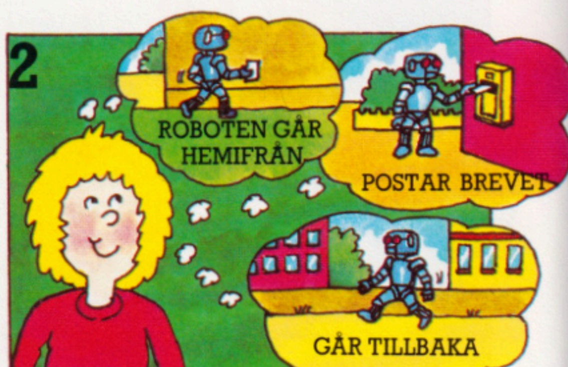
Att skriva program

Ett program är ungefär som spelregler eller ett bakrecept. Finns det ett fel i reglerna eller receptet blir spelet också fel eller kakan dålig. På samma sätt beror resultatet av datorns arbete på de instruktioner du har matat in. Innan du skriver ska du noga tänka igenom vad du vill att datorn ska göra.

Brev-program

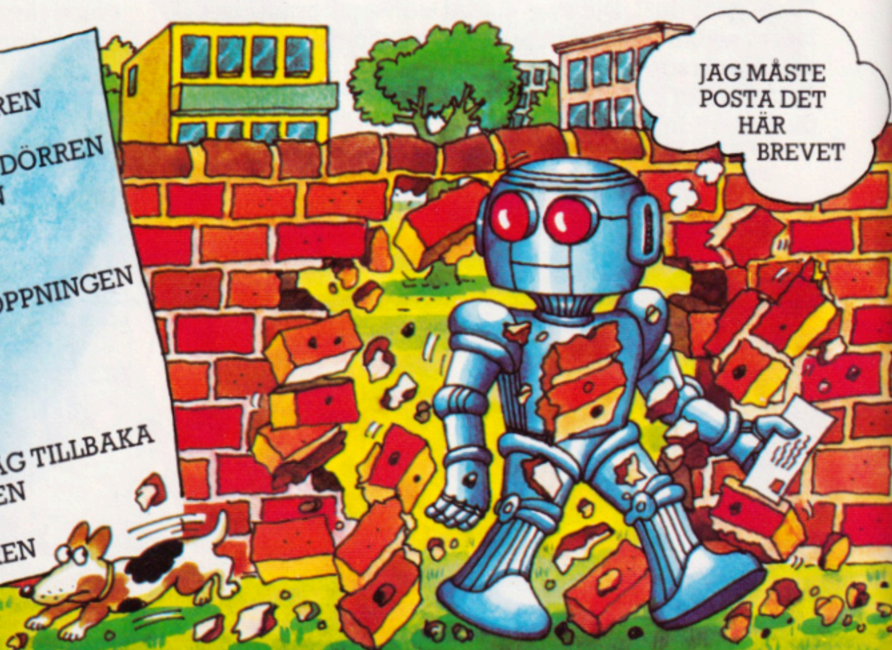


Du ska försöka skriva ett program som får en en robot att posta ett brev. En sådan instruktion som på bilden ovan är för svår för roboten..



Du måste tala om för roboten exakt hur den ska göra för att komma till brevlådan, posta brevet och sedan återvända hem.

3

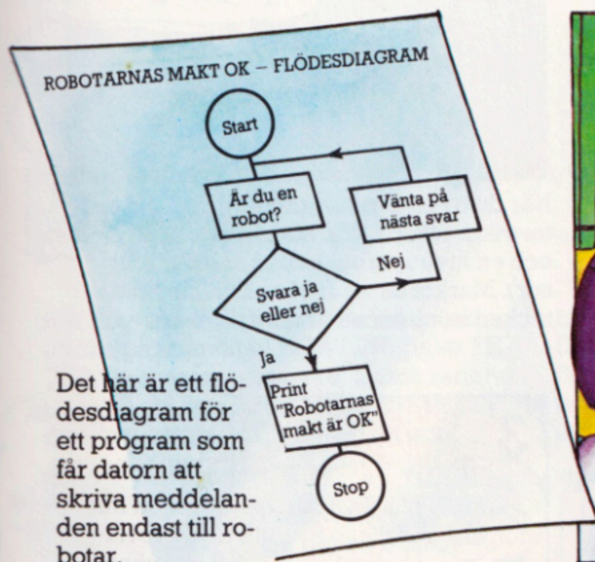


Inte ens instruktionerna på bilden ovan är tillräckligt detaljerade. De måste delas in i ännu mindre steg som datorn i roboten förstår.

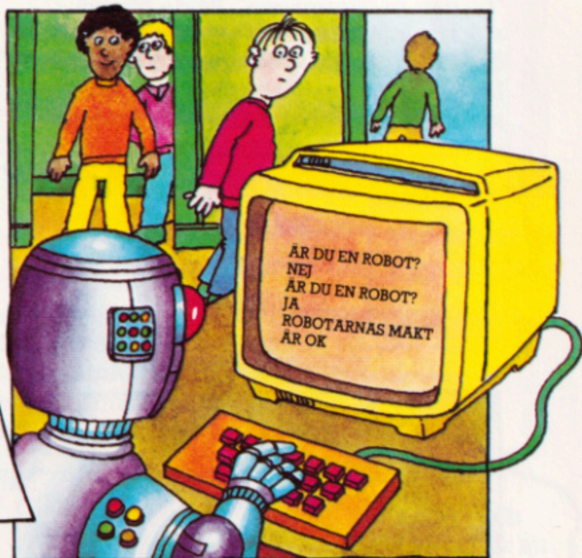
Roboten kommer att försöka genomföra dina instruktioner även om de är felaktiga eller ofullständiga. Fel i program kallas ofta bug och kan ibland leda till oväntade resultat.

Flödesdiagram

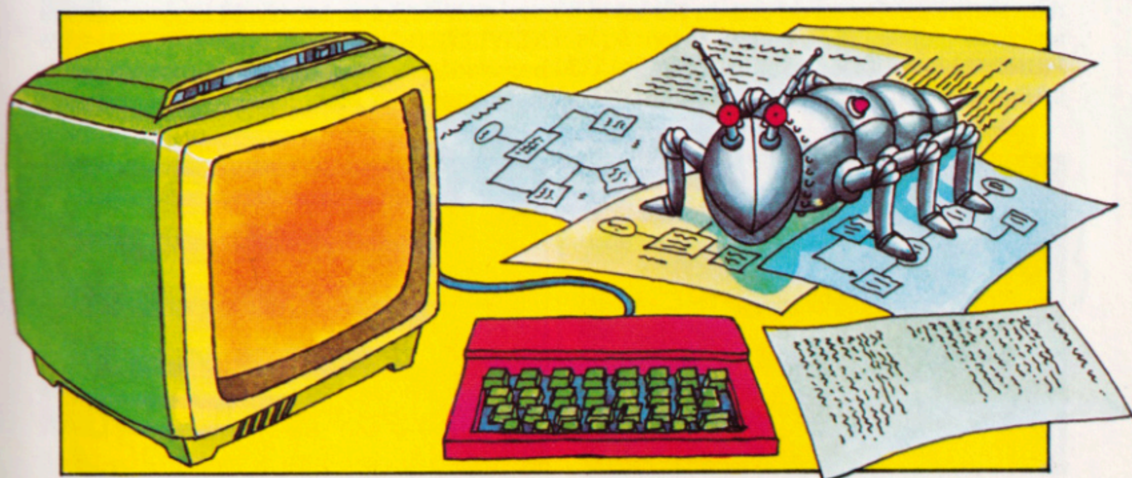
På bilden nedan ser du ett flödesdiagram. Rita ett sådant diagram över de viktigaste stegen som behövs för att lösa ditt problem. Diagrammet hjälper dig att se om du har tänkt rätt och att komma ihåg. Diagrammet visar i vilken ordning de olika stegen ska utföras, det kallas flödet i programmet.



Det här är ett flödesdiagram för ett program som får datorn att skriva meddelanden endast till robotar.



Ett flödesdiagram har rutor med olika utseenden för olika steg i programmet. Början och slutet av programmet finns i runda rutor. Vanliga instruktioner till datorn finns i rektangulära rutor, medan instruktioner där ett val beroende på den inmatade informationen måste göras finns i rombformade rutor. Linjer visar möjliga vägar som datorn kan ta.



Efter att ha bestämt alla detaljer som måste utföras kan du översätta instruktionerna till BASIC och testa programmet på din dator. Förmodligen kommer inte programmet att fungera med detsamma på grund av fel som du har gjort. Det kan vara stavfel som du gjorde när du matade in eller logiska fel i ditt program. För att få programmet att fungera som du har tänkt måste du hitta alla fel och rätta till dem (*). Det kan hända att ett fel gör att programmet ger ett annorlunda resultat än planerat, men ett resultat som du kanske ändå tycker om.

(*) På sid 42-43 finns några råd om hur du hittar fel.

Första ord i BASIC

Många ord i BASIC baseras på engelska ord och ibland är det lätt att gissa vad de betyder. PRINT t ex betyder "skriv ut på skärmen" eller skrivaren, RUN betyder starta programmet och INPUT betyder "ge datorn information". På de här två sidorna kan du se hur PRINT kan användas.
De flesta hemdatorer har en BASIC-tolk i sig från början och är därför direkt klara att programmeras i BASIC. (*)



När du trycker på tangenterna på din dator visas ofta på skärmen några tecken och en liten symbol kallad markör (cursor). Markören visar var nästa inmatade tecken kommer att hamna på skärmen.



För att få datorn att skriva ut ord på skärmen skriv PRINT med de ord du vill ha utskrivna inom citationstecken. T ex PRINT "SNIGLAR" för att få ordet SNIGLAR utskrivet på skärmen.



Datorn kommer inte att utföra någon instruktion förrän du har tryckt tangenten NEWLINE (eller RETURN eller ENTER - beroende på datorn) för att tala om att instruktionen är komplett.



Datorn kommer att visa på skärmen allt som du har skrivit mellan citationstecken. Det kan vara bokstäver, siffror, ord eller symboler. Observera att citationstecknen visas inte.



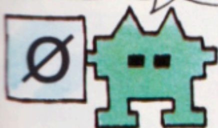
För att få tal utskrivna behövs inga citationstecken. För att sedan göra rent på skärmen kan du skriva CLS (eller annat kommando som används på din dator - kolla i handboken).

(*) På vissa datorer måste ett speciellt program matas in för att maskinerna skall förstå BASIC-språket.

Program skrivna i BASIC

I ett program inleds varje programrad med ett nummer. Det betyder att instruktionerna skall lagras i datorn och inte utföras förrän du säger till. På sidan 10 fanns inga nummer före instruktionerna, så datorn utförde den genast. Här har vi ett kort program som får datorn att visa symboler i form av ett ansikte.

På vissa datorer har siffran 0 ett streck över sig - så här:




```
10 PRINT "//////"
20 PRINT "I      I"
30 PRINT "I(.  .)I"
40 PRINT "I   -LI"
50 PRINT "VVVV"
60 END
```

Radnumreringen sker i jämna tiotal så du kan lägga extra instruktioner emellan utan att behöva numrera om hela programmet.



Det är inte alla datorer som behöver den här raden.

När du skriver in ditt program måste du trycka på tangenten NEWLINE (eller RETURN) i slutet av varje rad. Dessa rader visas på skärmen men utförs inte förrän du skriver RUN. Se upp med att inte blanda bokstaven O med siffran 0 (noll). Det kan leda till fel i ditt program. De flesta datorer har RUBOUT eller DELETE tangent som du använder för att ta bort ett felaktigt tecken på raden.

```
RUN
//////
I    I
I(..)I
I – LI
VVVV
```

Datorn visar allt du har skrivit mellan citationstecken, även mellanslag.

```
RUN
FEL"
MEDDELANDE
```

Felmeddelande

Skriv LIST (och NEWLINE) för att få programmet utskrivet.

När du har skrivit in alla rader ska du kontrollera dem noga för att se att det inte finns några fel. Sedan kan du skriva RUN (och NEWLINE) för att tala om för datorn att den ska utföra (exekvera) ditt program.

Om programmet inte alls fungerar eller om resultatet av en körning inte blir vad du har tänkt måste du skriva ut programmet på nytt så att du kan hitta felet. Skriv LIST (och NEWLINE). Det kan hända att datorn själv talar om vad som är fel.

1 Rättelseprogram

```
RUN
FEL"
MEDDELANDE
10 PRINT "//////"
20 PRINT "I      I"
30 PRINT "I(.  .)I"
40 PRINT "I   –LI"
50 PRINT "VVVV
60 END
```

Citationstecken fattas

2

```
RUN
FEL"
MEDDELANDE
10 PRINT "//////"
20 PRINT "I      I"
30 PRINT "I(.  .)I"
40 PRINT "I   –LI"
50 PRINT "VVVV
60 END
50 PRINT " VVVV "
```

Raden inmatad igen, rätt.

I de flesta fall talar datorn om vad du har gjort för fel. Felmeddelanden finns beskrivna i handboken som hör till datorn. Det enklaste sättet att rätta till felet är att skriva hela raden på nytt. Datorn ersätter då den gamla raden med den nya. För att få bort en hel rad från programmet skriver du radnummer och NEWLINE. På olika datorer finns det olika sätt att rätta en del av en rad, t ex genom att använda EDIT eller COPY funktioner. Detta beskrivs också i handboken.



Övningsuppgift Försök att ändra programmet så att det ger ett annat resultat

Att mata datorn med data

För att få datorn att göra någonting nyttigt och inte bara visa saker på skärmen måste du ge den material, "data" att arbeta med.

1

```
10 LET A = 6
20 LET B = 7
30 LET C = 23
40 LET D = 4
```

De här talen skall lagras i datorns minne.

Det här är namn för fack i minnet.

Du måste sätta en etikett (namn) på varje fack i minnet där du lagrar data. Det är nödvändigt för att du senare ska kunna få tag i det du har lagrat. Som namn kan du använda vanliga bokstäver. För att markera ett fack i minnet och lagra ett tal där kan du använda ordet LET, som på bilden ovan. Ett sådant utrymme kallas för variabel då innehållet kan variera vid olika tillfällen.

2

```
10 LET A=3
20 LET A$="SNIGLAR"
30 LET B=43
40 LET B$="ROBOTAR"
```

Glöm inte citationstecken

Använd olika namn för bokstäver och symboler som du skall lagra i minnet. En rad eller del av rad med bokstäver och symboler kallas för teckensträng, och du använder bokstäver tillsammans med ett dollartecken (eller en "sol") för att markera dessa.
En teckensträng lagras i minnet med LET på samma sätt som ett tal, men bokstäver och symboler måste omslutas av citationstecken som på bilden ovan.

3

```
10 LET B=365
20 LET D$="DAGAR PER ÅR"
30 LET L$="UNDANTAGET SKOTTÅR"
40 PRINT B
50 PRINT D$
60 PRINT L$
70 END
```

Inte alla datorer behöver END.

4

```
RUN
365
DAGAR PER ÅR
UNDANTAGET SKOTTÅR
```

För att visa informationen på skärmen skriver du ordet PRINT tillsammans med variabelnamn, t ex PRINT A$. Det här korta programmet skriver ut data från variablerna B, D$ och L$.

Du kan köra programmet så många gånger du vill. Varje gång kommer datorn att skriva ut samma information. Data i variablerna ändras inte förrän du ändrar det.

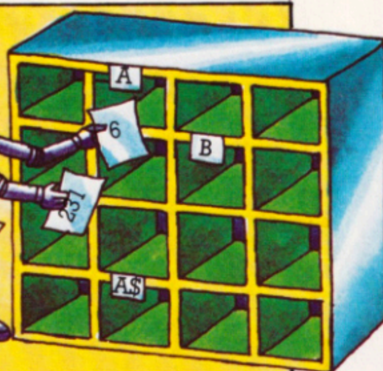
Ett annat sätt att lagra information

```
10 READ A
20 READ B
30 READ A$
40 DATA 6,231,FREDAG
```

Du måste ha rätt sorts namn på tal respektive bokstäver

Kommatecknen

Vissa datorer måste ha dataord inom citationstecken.

Ett annat sätt att lagra data är att använda orden READ och DATA, som ovan. READ-raden talar om för datorn att sätta ett namn på minnesutrymmet medan DATA-raden innehåller informationen.

När du kör programmet lagras data i minnet i den ordning de står i programmet. Kommatecken används för att dela in data i mindre block (*).

Några program

1

```
10 READ Q
20 READ X$
30 DATA 24, BIG MAC
40 PRINT Q
50 PRINT X$
60 END
RUN
24
BIG MAC
```

Komma

Det här är en sk datasats. Kommatecknet ingår.

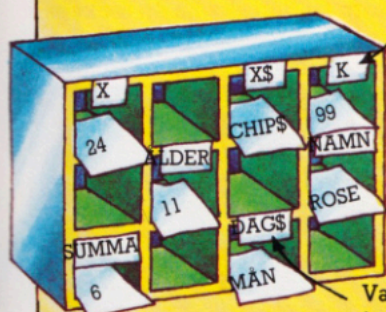
2

Citat

```
10 LET A$="ROBOTAR ÄR KUL"
20 LET B$="OM DU GILLAR"
30 LET C$="STÅLIDIOTER"
40 PRINT A$
50 PRINT B$
60 PRINT C$
70 END
RUN
ROBOTAR ÄR KUL
OM DU GILLAR
STÅLIDIOTER
```

Det här är två program. I det ena används READ och DATA, i det andra används LET, för lagring av data i datorns minne.

Mera om variabler



Variabelnamn för tal

Variabelnamn för teckensträng.

Här finns några ord som inte får användas som namn på variabler eftersom de är BASIC-ord.

LIST$

NEWS$

GOSUB

Variabler är fack i minnet med namn för lagring av information (data). En variabel som innehåller tal kallas för talvariabel och en med bokstäver kallas för strängvariabel. Innehållet i dessa variabler kan ändras under programmets exekvering. På vissa datorer kan vanliga ord användas som namn för variabler, men inte ord som ingår i BASIC, då datorn skulle få problem.

* Denna metod används ej på ZX81-datorer

Att använda INPUT

Ett annat sätt att mata en dator med data är att använda ordet INPUT. Med det kan du ge datorn information medan programmet "går" och det kan vara olika varje gång du matar in den.



INPUT används med t ex A som namn för tal och A$ för strängar. När datorn stöter på ordet INPUT i programmet sätter den ett namn på ett ledigt minnesutrymme och ber dig mata in det som ska lagras där. Vanligen visas då ett frågetecken eller någon annan symbol på skärmen. När du har matat in data lagras den i minnet och datorn går vidare i programmet.

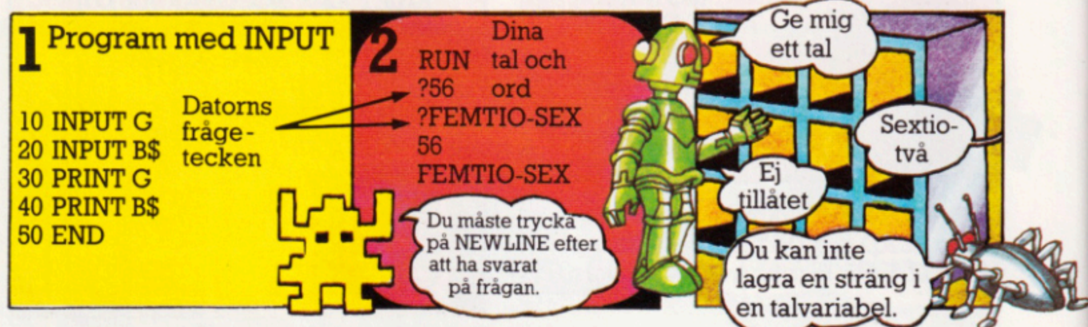


Bild 2 visar vad som händer när du kör ditt program. När datorn träffar på ordet INPUT i rad 10 skriver den ut ett frågetecken på skärmen och väntar på att du ska skriva in ett tal till variabel G. Sedan skriver den ut ett nytt frågetecken för INPUT i rad 20. Denna gång måste du mata in ord eller symboler därför att B$ säger datorn att den ska ha en teckensträng.



Om du har en dator så kan du mata in programmet och skriva RUN för att starta det. När datorn ber dig att mata in någonting kan du svara med ditt namn och åldern, eller också kan du svara med ett påhittat namn och åldern som på bilden ovan. Försök flera gånger med olika svar. Starta med RUN. Varje gång kommer datorn att skriva ut exakt samma sak som du har skrivit in i N$ och A.

Program som skriver poesi

Nu kan du så mycket BASIC att du kan skriva en dikt med hjälp av din dator. Det kan du göra med det här programmet som använder PRINT och INPUT.

```
10 PRINT "VAD HETER DU"
20 INPUT N$
30 PRINT "DIKT AV"
40 PRINT N$
50 PRINT "SKRIV ETT ORD"
60 PRINT "SOM RIMMAR PÅ FRI"
70 INPUT A$
80 PRINT "HÄR ÄR DIKTEN"
90 PRINT "FRÅN DATORRÄDSLA ÄR
    JAG FRI"
100 PRINT "LYCKLIG SOM ETT"
110 PRINT A$
120 END
```

Här skrivs ditt namn

På denna rad skrivs ordet





Skriv RUN för att pröva igen med något annat rimord.

Programmet får datorn att fråga dig om ditt namn, som sedan lagras i N$ och skrivs ut på rad 40. Ordet som du väljer lagras i A$ för att sedan skrivas ut på rad 110 som en del av dikten. Har du en dator så kan du starta programmet flera gånger med olika ord som skrivs in när datorn kommer till rad 70.

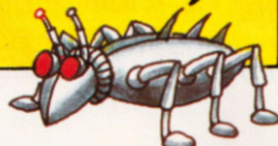
En övningsuppgift

Kan du skriva ett program som frågar om ditt namn och sedan skriver ut ett hej som följs av ditt namn och ett meddelande?

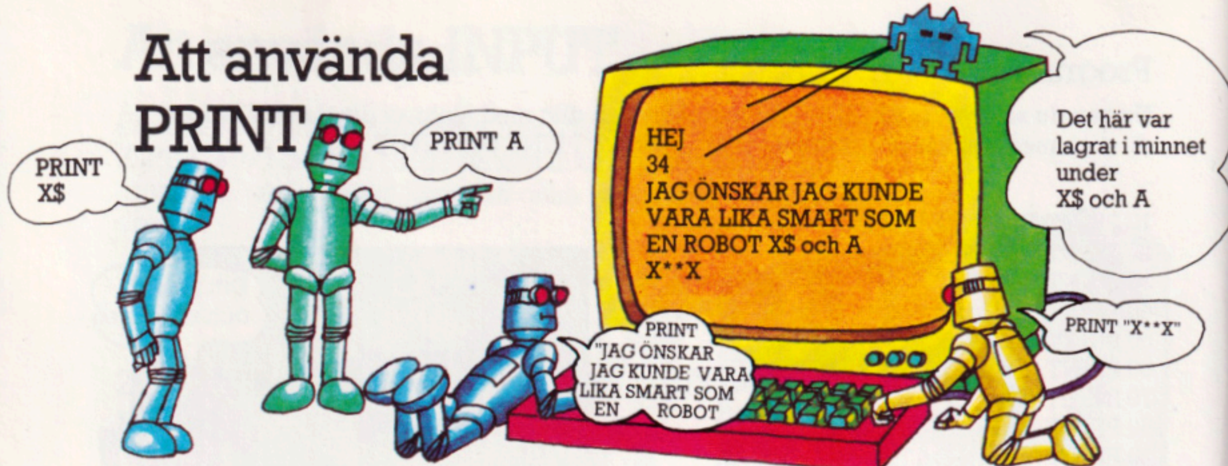
En komihåg-lista för inmatning av program

1. Innan ett nytt program matas in skriv NEW. Detta rensar minnet från gamla program och variabler.
2. Kom ihåg att avsluta varje rad med NEWLINE (eller RETURN, beroende på datorn).
3. När hela programmet är inmatat måste du kontrollera att det inte finns några stavfel. Kontrollera också att alla rader finns med.
4. Sedan kan du rensa skärmen (skriv CLS eller det ord som används på din dator). Skriv RUN för att starta programmet.
5. För att få se programmet igen skriver du LIST. För att titta på en speciell rad, skriver du LIST och radnummer (det kan vara litet olika på olika datorer).
6. För att stoppa ett program trycker du på ESCAPE eller BREAKE (kontrollera först i handboken, för på vissa datorer raderar ESCAPE bort hela programmet). Vill du starta igen skriv bara RUN.

På sid 42–43 finns några goda råd för att hjälpa dig hitta fel.



Att använda PRINT



Hittills har du sett hur man använder ordet PRINT för att visa ord och tal samt innehållet i variablerna på skärmen. Nedan kan du se hur komma och semikolon kan användas för att forma texten på skärmen. Med PRINT kan du få datorn att utföra beräkningar. Du ser längst ned på sidan hur det går till. På sidan bredvid visar vi vad du kan göra med variablerna.

Komma och semikolon

```
10 PRINT "JAG ÄR",
20 PRINT "UPPDELAD"
```
Komma → JAG ÄR UPPDELAD

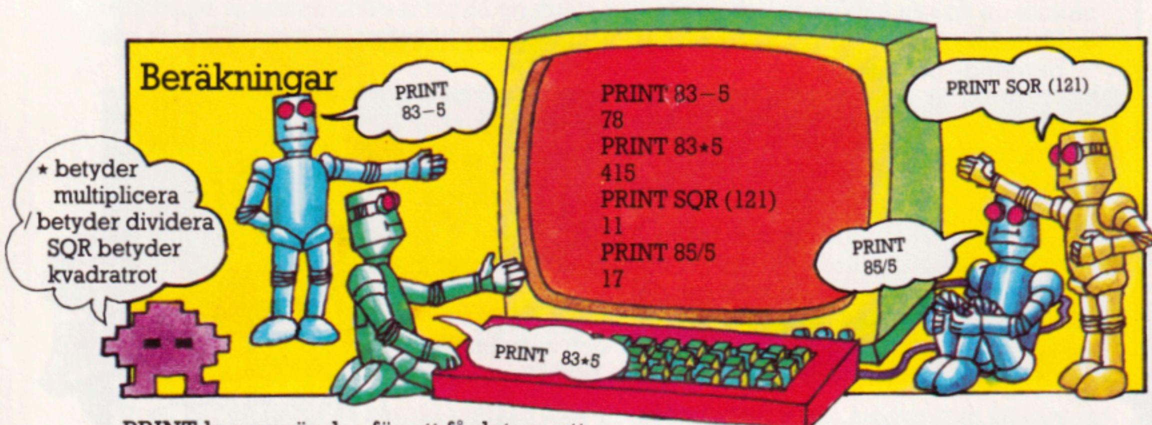
```
10 PRINT "JAG ÄR";
20 PRINT "IHOPTRYCKT"
```
Semikolon → JAG ÄR HOPTRYCKT

```
10 PRINT "JAG ÄR VERKLIGEN"
20 PRINT
30 PRINT "UPPDELAD"
```
→ JAG ÄR VERKLIGEN

UPPDELAD

Det blir en tom rad vid utskriften när ordet PRINT står på en egen rad i programmet.

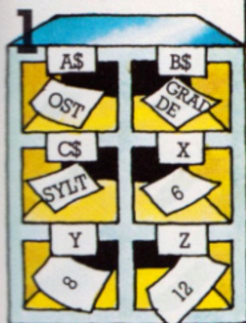
De här raderna visar hur du kan använda komma och semikolon för att tala om för datorn var nästa tecknet skall skrivas ut. Komma betyder flytta en bit, medan semikolon betyder stanna där du är. Bilden ovan visar hur utskriften blir. Om ordet PRINT står på en egen rad i programmet gör datorn en tom rad vid utskriften.



PRINT kan användas för att få datorn att göra beräkningar direkt på skärmen. För addition och subtraktion används vanliga symboler, för multiplikation används * och / för division.

Datorn kan också göra svårare beräkningar som sinus, cosinus, kvadratrötter (SQuare Roots) osv.

Mer om variabler



På de flesta datorer måste du skriva in ett mellanslag (space) på sidorna om en variabel och innanför citationstecken.

2

Mellanslag

```
PRINT "JAG ÅT ";X;" SMÖRGÅSAR MED JORDNÖTSSMÖR OCH ";A$;
JAG ÅT 6 SMÖRGÅSAR MED JORDNÖTSSMÖR OCH OST

PRINT "JAG ÅT ";Z;" SMÖRGÅSAR MED ";C$
JAG ÅT 12 SMÖRGÅSAR MED SYLT
```

Att skriva ut enbart variabler ger inte så mycket. Skriv också en text som förklarar vad varje variabel betyder. För att skriva ut orden och variablerna tillsammans, måste orden skrivas inom citationstecken, medan variablerna skrivs med semikolon på båda sidor. Vill du flytta isär orden kan du använda komma istället för semikolon.

3

```
LET X=X+1

LET C$=C$+"OCH OST"
```



Med hjälp av programmet kan du ändra innehållet i minnet som du ser ovan. För datorn betyder dessa instruktioner att den ska addera 1 till värdet i X och addera bokstäverna "OCH OST" till innehållet i C$.

4

Mellanslag

```
PRINT "JAG ÅT ";X;" SMÖRGÅSAR
MED ";C$
JAG ÅT 7 SMÖRGÅSAR MED SYLT
OCH OST
```

Nästa gång du ber din dator att skriva ut variablerna kommer nya ord och tal att visas.

5

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 END
RUN
63
1.28571
```

Multiplicera

Dividera

Du kan göra beräkningar med variabler. Datorn tar fram talen från minnet och bearbetar dem efter dina önskemål.

Övningsuppgift

1. Skriv ett program som adderar tal till variablerna i programmet här till vänster, så att det skriver ut 100 och 1 på samma rad med litet utrymme mellan.

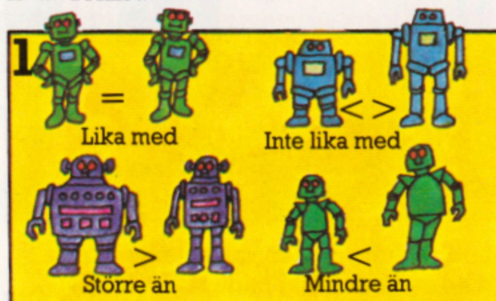
2. Ändra rad 30 och 40 så att datorn skriver ut talen, vad du gör med dem och resultatet, t ex "7 gånger 9 är lika med 63".

3. Ändra ditt svar i övningsuppgiften på sidan 15 så att ditt namn och meddelandet skrivs ut på samma rad.

Hur datorer jämför saker

En av de mest användbara saker en dator kan utföra är att jämföra olika informationer och sedan göra något i anslutning till det funna resultatet. För att få fram detta använd orden IF ... THEN.





En dator kan göra jämförelser på flera olika sätt. Symboler för tester ser du ovan. Datorn kan kontrollera om innehållet i två variabler är lika, olika eller om någon är större eller mindre än den andra.

2

```
IF A=B THEN PRINT "DE ÄR LIKA"
IF A>B THEN PRINT "A ÄR STÖRRE"
IF A<B THEN PRINT "A ÄR MINDRE"
IF A<>B THEN PRINT "DE ÄR INTE
LIKA"
```

De här raderna visar hur du kan använda IF och THEN för att datorn skall jämföra innehållet i två variabler. En sådan jämförelse kan göras på alla typer av data - ord, tal och variabler, dvs minnesinnehållet också.

3 Väderprognos

```
10 PRINT "HUR ÄR VÄDRET IDAG"
20 INPUT W$
30 IF W$="REGN" THEN PRINT "TA
PARAPLYET"
40 IF W$="SOL" THEN PRINT "BRA"
50 END
```

Svarar du med dessa ord händer ingenting.

VARMT SNÖ

4

```
RUN
HUR ÄR VÄDRET
IDAG
? SOLIGT
BRA
RUN
HUR ÄR VÄDRET
IDAG
? REGNIGT
TA PARAPLYET
```

Här har du ett program som använder IF och THEN. I rad 20 lagras ordet du matade in i variabeln W$. Sedan, i raderna 30 och 40, kontrollerar datorn att ordet i W$ är samma som "regn" eller "soligt". Stämmer det får du ett svar. Har du skrivit in ett annat ord i rad 20 händer ingenting. Men du kan också ändra orden i rad 30 och 40 och sedan försöka igen.

5 Åldern

```
10 PRINT "HUR GAMMAL ÄR DU"
20 INPUT A
30 IF A>16 THEN PRINT "GAMMAL"
40 IF A<16 THEN PRINT "UNG"
50 IF A=16 THEN PRINT "PERFEKT"
RUN
HUR GAMMAL ÄR DU
?16
PERFEKT
```

6 Lektion i franska

```
10 PRINT "HUR SÄGER MAN RÖD PÅ FRANSKA"
20 INPUT A$
30 IF A$="ROUGE" THEN PRINT "KORREKT"
40 IF A$><"ROUGE" THEN PRINT "NEJ, ROUGE"
RUN
HUR SÄGER MAN RÖD PÅ FRANSKA
?BLEU
NEJ ROUGE
```

I ålder-programmet här ovan jämför datorn det inmatade data A med värdet 16. Är data större än 16 skriver den GAMMAL. Är det mindre skriver den UNG, är det precis 16 skriver den PERFEKT. I det andra programmet skriver datorn ut ett av svaren, beroende på om A$ är lika med "rouge" eller inte.

Övningsuppgift
Kan du skriva ett program som frågar dig om beräkningsresultat och sedan svarar "korrekt" eller ger dig de rätta svaren?

Program som innehåller hopp

1

```
IF A=6 THEN LET A$="SEX"
IF X=Y-2 THEN LET Z=0
IF S=T THEN STOP
IF R<10 THEN GOTO 30
```



Du kan ge datorn nästan vilken instruktion som helst efter ordet THEN, vilket visas ovan. Du kan bestämma vilken rad datorn ska utföra om villkoret är uppfyllt. Du skriver GOTO följt av ett radnummer.

2

```
10 INPUT K$
20 IF K$="JA" THEN GOTO 100
30 IF K$="NEJ" THEN GOTO 200

100 PRINT "DU SKREV JA"
110 STOP.

200 PRINT "DU SKREV NEJ"
210 END
```



På en del datorer räcker det med radnumret. Ofta behövs också ordet STOP, för att hindra att datorn går genom programmet ett oändligt antal gånger.

Program som räknar

```
10 PRINT "SKRIV IN ETT TAL"
20 INPUT A
30 PRINT "SKRIV IN ETT ANNAT TAL"
40 INPUT B
50 PRINT "VILL DU ADDERA"
60 PRINT "SUBTRAHERA, MULTIPLICERA"
65 PRINT "DIVIDERA ELLER SLUTA"
70 INPUT C$
80 IF C$="ADDERA" THEN PRINT A+B
90 IF C$="SUBTRAHERA" THEN PRINT
A-B
100 IF C$="MULTIPLICERA" THEN
PRINT A*B
110 IF C$="DIVIDERA" THEN PRINT A/B
120 IF C$="SLUTA" THEN STOP
130 GOTO 10
```

```
RUN
SKRIV IN ETT TAL
?17
SKRIV IN ETT ANNAT TAL
?184
VILL DU ADDERA
SUBTRAHERA, MULTIPLICERA
DIVIDERA ELLER SLUTA
?ADDERA
201 <---------- Datorns svar
SKRIV IN ETT TAL
?
```



I det här programmet lagras de tal du skriver in i A och B och din instruktion i C$. I raderna 80 till 120 jämför datorn dina svar med fem olika ord. När det rätta ordet hittas utförs instruktionen. Alla rader med fel ord hoppas över.

Gissa ålder-programmet

1

```
10 PRINT "GISSA MIN ÅLDER"
20 INPUT G
30 IF G><14 THEN PRINT
   "FÖRSÖK IGEN"
40 IF G><14 THEN GOTO 20
50 PRINT "KORREKT"
60 END
```

2

```
RUN
GISSA MIN ÅLDER
?15
FÖRSÖK IGEN
?14
KORREKT
```

3

```
GISSA MIN ÅLDER
?15
YNGRE ÄN SÅ
?13
ÄLDRE ÄN SÅ
?14
KORREKT
```



Det här programmet kommer att upprepas tills G=14. När G=14 kommer datorn att hoppa över raderna 30 och 40 och skriva ut korrekt. Kan du ändra programmet så att du får ledtrådar, som på bilden ovan?

Program med mycket BASIC

På följande två sidor används de flesta BASIC-ord som hittills har förekommit i boken. Det första programmet är ett rymdspel för två. Har du ingen dator studera programmet och försök förstå hur det fungerar.

Rymdspel

```
10 PRINT "FIENDER FINNS INOM
   KOORDINATERNA A"
20 INPUT A
30 "OCH KOORDINATERNA B"
40 INPUT B
50 CLS
60 PRINT "RYMDSTYRKORNA FINNS
   INOM KOORDINATERNA C"
70 INPUT C
80 PRINT "OCH KOORDINATERNA D"
90 INPUT D
100 CLS
110 LET X=SQR((A−C)*(A−C)+(B−D)*(B−D))
120 PRINT "NI ÄR NU"
130 PRINT X;"RYMDENHETER FRÅN VARANDRA"
140 IF X<1,5 THEN PRINT "FIENDEN HITTAD"
150 IF X<1,5 THEN STOP
155 PRINT "VAD ÄR ERA NYA POSITIONER"
160 GOTO 10
170 END
```

Raderna 10 till 40 lagrar fiendens koordinater i A och B

Rad 50 rensar skärmen

Rad 60 till 90 lagrar styrkornas koordinater i C och D

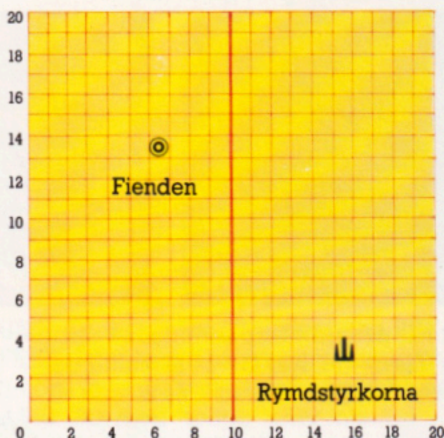
Denna rad räknar ut hur långt från varandra de finns

Om X är mindre än 1,5 stannar programmet, om X är större upprepas det

I det här spelet är en person fienden och den andra spelar rymdstyrkorna som försöker fånga fienden. Varje spelare ritar en hemlig karta där de markerar sina positioner (hur detta görs kan du se nedanför). Datorn får koordinater för dessa rutor och kan sedan räkna ut hur långt från varandra ni är. Spelare kan använda datorns beräkningar för nästa drag.

Hur man spelar

För sin hemliga karta ritar varje spelare ett rutnät med 20 rutor som numreras som på bilden bredvid. Fienden startar från den vänstra kanten och rymdstyrkorna från den högra. För varje drag kan de flytta två rutor - upp, ner, i sidled eller diagonalt - och sedan uppge sina nya positioner. När de är mindre än 1,5 rymdenheter (dvs rutor) från varandra har rymdstyrkorna fångat fienden.



Hur du kan få datorn att verka intelligent.

I det här programmet får du datorn att svara på dina svar på datorns frågor. Hur programmet arbetar kan du se på bilderna längst ner på sidan. INPUT används här på ett annorlunda sätt, vilket gör att programmet blir kortare och lättare att läsa och förstå.

1

```
10 INPUT "SKRIV IN ETT TAL";N
20 INPUT "OCH ETT TILL";M
30 PRINT N;" GÅNGER ";M;"
    ÄR LIKA MED";N*M
```

BBC's mikrodator behöver inte semikolon

2

```
RUN
SKRIV IN ETT TAL?10
OCH ETT TILL?8
10 GÅNGER 8 ÄR 80
```

På en ZX81 måste du skriva
10 PRINT "SKRIV IN ETT TAL"
15 INPUT N

På de flesta datorer (men inte ZX81) kan du göra INPUT-raden tydligare genom att skriva orden före variabelnamnen inom citationstecken. När programmet körs kommer ett frågetecken efter INPUT-radens text.

Program

```
5 LET C=0
10 PRINT "JAG SKULLE VILJA PRATA MED DIG"
20 INPUT "BERÄTTA FÖR MIG OM ALLT
   SOM HÄNDE DIG DENNA VECKA":A$
30 READ B$
40 PRINT B$;
50 INPUT C$
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA VARFÖR, VARFÖR DET
95 DATA KAN DU FÖRKLARA, VARFÖR
98 DATA KAN DU SÄGA VARFÖR, VAD VAR ORSAKEN
100 PRINT "SÅ ANLEDNINGEN ATT DU"
110 PRINT " ";A$
120 PRINT "FANNS EGENTLIGEN I DITT SVAR"
130 PRINT " ";C$
140 PRINT "VA' KONSTIGT"
150 PRINT "STARTA MIG IGEN SÅ FÅR VI VETA MERA"
160 END
```

Rad 40: Detta gör att datorn stannar på denna rad.

Rad 50: Ditt svar är lagrat i C$.

Rad 10–20: Det här är ett nytt sätt att använda INPUT. Ditt svar lagras i A$

Vid rad 30 letar datorn efter första raden med DATA, tar första delen och lagrar den i B$

Variabel C i rad 60 och 70 är en räknare. Den håller reda på hur många gånger programmet repeteras. När C=6 då har alla DATA-delar använts och datorn går vidare till rad 100.

Rad 80 gör att datorn går tillbaka till rad 30 och ersätter data i B$ med nästa delen från DATA-raden.

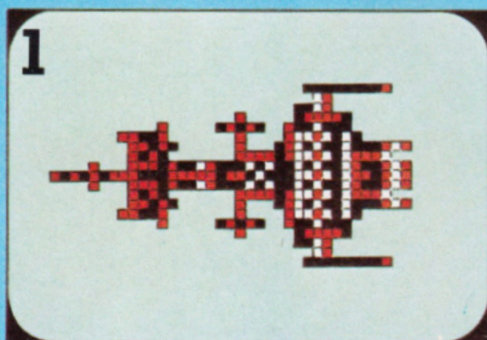
Den tomma platsen på rad 110 och 130 gör att det blir ledigt utrymme för dina svar på skärmen.

Hur fungerar det

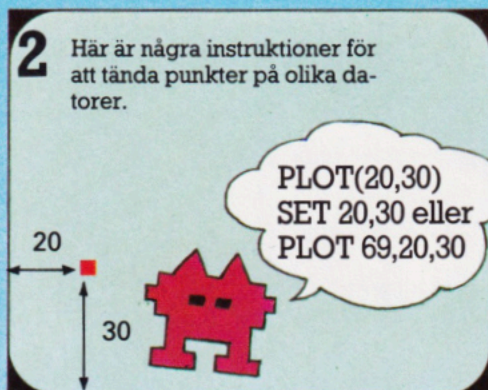
Rita bilder

En dator ritar bilder genom att lysa upp små fyrkanter på skärmen. Varje fyrkant kallas för punkt och varje punkt behöver en separat instruktion från datorn för att tändas eller släckas. En del datorer kan också få punkter i olika färger.

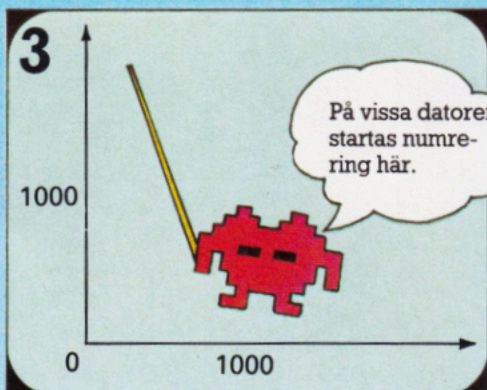
På dessa två sidor kan du se hur det går till att rita enkla bilder på skärmen med hjälp av BASIC. Instruktionerna som visas här ger bara svart-vita bilder.



Du kan faktiskt se punkter i en vanlig datorbild. En dator med mycket stort minne kan göra bilder med hjälp av tusentals mycket små punkter. Dessa bilder kallas för grafik med hög upplösning.



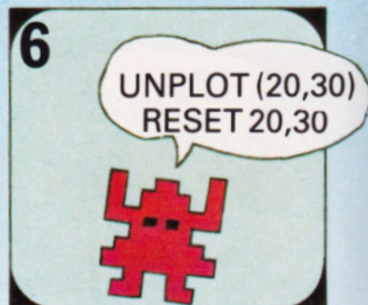
Instruktionen för att tända punkter varierar för olika datorer, men är vanligen någonting i stil med PLOT(X,Y). X och Y är punktens koordinater (X horisontellt och Y vertikalt).



En dator som har grafik med hög upplösning kan rita bilder som är 1000 gånger 1000 punkter. En mindre dator har kanske 60 gånger 40 som mest. (Kontrollera hur många punkter din dator har så att du inte hamnar utanför det grafiska området när du skall rita bilder.)



Bilder som görs av datorer kallas vanligen grafiska. Vissa datorer behöver en speciell instruktion innan du kan se dina bilder på skärmen. BBC's micro t ex vill ha ordet MODE med ett tal.



En punkt kan också släckas med hjälp av en instruktion som UNPLOT (X,Y). För program i den boken använder vi PLOT och UNPLOT. Se efter i handboken vad instruktionerna heter för din dator.

Ritprogram

PLOT-instruktioner är olika för olika datorer

1

```
10 PRINT "SKRIV IN
   TVÅ TAL"
20 INPUT X
30 INPUT Y
40 PLOT (X,Y)
50 GOTO 10
```



Du måste trycka RETURN (eller NEWLINE) efter varje tal.



2

```
RUN
SKRIV IN TVÅ TAL
?24        ← första punkten
?24
SKRIV IN TVÅ TAL
?30
?15        ← andra punkten
```

Det här korta programmet frågar efter två tal och tänder sedan en punkt med dessa tal som koordinater. Om du testar programmet på din dator så kontrollera att de tal du matar in ligger inom det grafiska området.

Rad 50 gör att programmet upprepas ett oändligt antal gånger. Det enda sättet att stoppa det är med BREAK (eller något annat speciellt ord). Du kan också lägga in en räknare för att få programmet att sluta efter t ex 6 gånger.

Att rita en bild

```
10 LET X=10
20 LET Y=10
30 PLOT (X,Y)
40 LET X=X+1
50 LET Y=Y-1
60 IF X<14 THEN GOTO 30
```

Det här programmet ritar en diagonal linje som går nedåt.

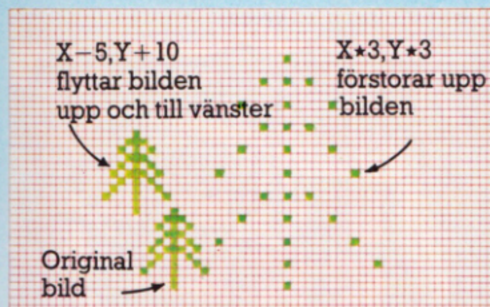
```
100 LET Y=Y+1
110 LET X=X+1
120 PLOT (X,Y)
130 IF X<20 THEN GOTO 100
```

Här ritas en diagonal linje som går uppåt.

Genom att 1 adderas till X men inte till Y ritas en horisontell linje.

Genom att 1 adderas till Y men inte till X ritas en vertikal linje.

Först måste du rita upp bilden på ett rutat papper och räkna ut koordinaterna. Sedan kan du göra ett program som ritar alla rutor. Genom att mata in startvärdena i X och Y och sedan addera eller subtrahera från dem och genom att få programmet att upprepa sig självt kan du få datorn att rita en följd av punkter som på bilden ovan.



Efter det att programmet har skrivits är det ganska lätt att ändra bilden genom att ändra talen. Du kan flytta den till olika delar av skärmen genom att ändra startvärden, eller förstora den genom att multiplicera alla tal med tre.

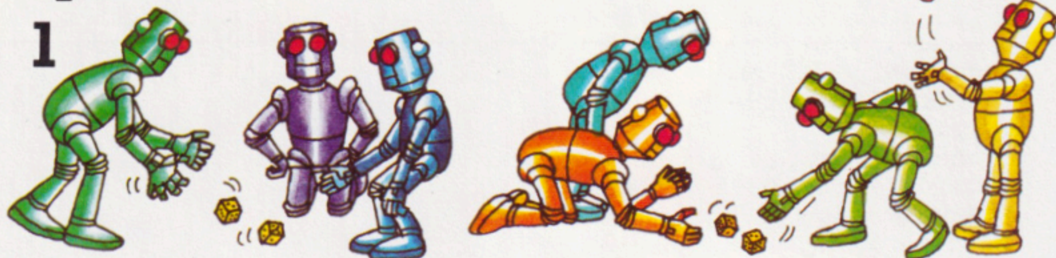
Ett annat sätt



Det är bara mycket enkla bilder som du kan göra med PLOT. För att göra mera komplicerade bilder behöver du speciella verktyg, som t ex ett digitaliseringsbord. Du kan då placera din bild på bordets yta och följa bildens konturer med hjälp av ett speciellt föremål, kallad "puck". Bildens koordinater matas då automatiskt in i datorn.

Övningsuppgift – Kan du göra ett program som ritar dina initialer på skärmen? Det finns ett exempel på sidan 44.

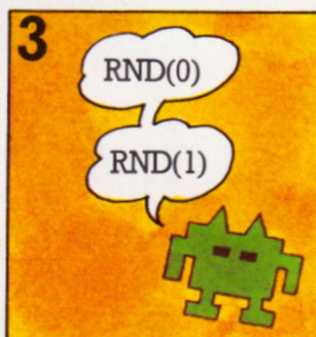
Spel



När du kastar ett par tärningar kan du inte förutse vilka nummer som kommer upp. Alla sex nummer har samma chans. Med hjälp av en dator kan du producera oförutsebara tal – slumptal.

En dator innehåller ett speciellt program för generering av slumptal. Ibland kommer samma tal flera gånger, men i en längre serie kommer alla tal ungefär lika ofta.

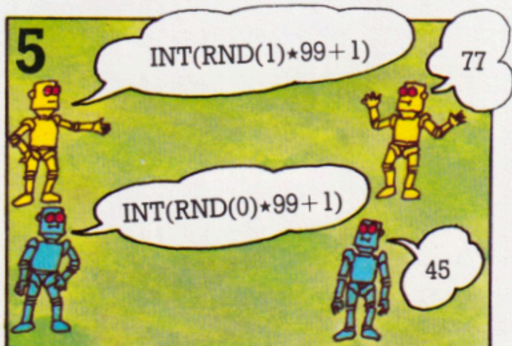


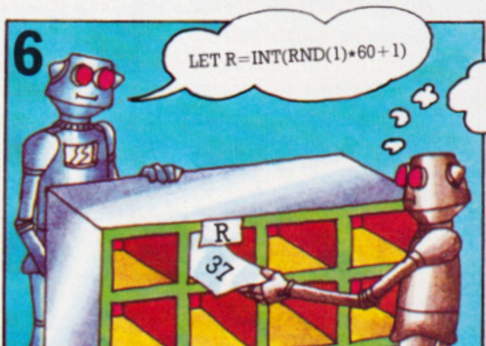




Vill du att datorn ska ta fram ett slumptal skriver du ordet RND. Vissa datorer vill ha 1 eller 0 inom parentes efter ordet. Titta i din handbok efter den rätta instruktionen.

RND-instruktionen ger alltid ett tal som är mindre än ett. På vissa datorer kan du skriva ett tal efter RND, t ex RND(99). Då tar datorn fram ett slumptal som ligger mellan 1 och talet inom parentesen.





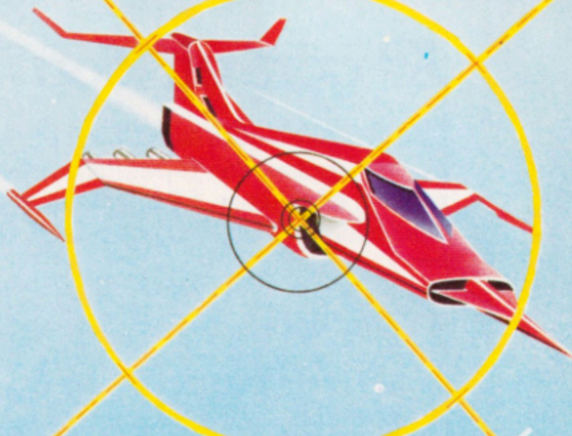
På några datorer måste du använda ordet INT (integer, dvs ett heltal) följt av RND-instruktion, RND(0) eller RND(1), beroende på datorn. Sedan får du multiplicera med något högt tal och addera ett så att resultatet blir större än 1.

Den här instruktionen betyder ta fram ett slumptal och lagra det i variabeln R. I program i den här boken använder vi instruktionen INT(RND(1)*60+1) för att få fram ett tal mellan 1 och 60. För vissa datorer måste denna instruktion ändras litet.



Övningsuppgift – Vet du hur man gör för att få fram ett slumptal mellan 10 och 20?

Rymdattack



Det här är ett program för ett spel som använder slumptal. I spelet befinner du dig på ett rymdskepp som blir anfallet av fientliga styrkor. Skeppets dator lokaliserar fienden och meddelar dig deras position. För att träffa fiendens enheter måste du räkna ut avståndet till dem genom att multiplicera koordinaterna och mata in svaren i datorn.

```
10 LET C=0
20 LET A=INT(RND(1)*20+1)
30 LET B=INT(RND(1)*20+1)
40 PRINT "FIENDENS RYMDSKEPPS -
   KOORDINATER"
45 PRINT "ÄR ";A,B;" GE ELD"
50 INPUT X
60 LET C=C+1
70 IF X=A*B THEN PRINT "FIENDENS SKEPP
   FÖRSTÖRT"
80 IF X<>A*B THEN PRINT "MISSAT"
90 IF C<6 THEN GOTO 20
100 END
```

C är en räknare som räknar hur många gånger programmet har repeterats. För varje gång adderas 1 till C.

De här två raderna producerar slumptal för fiendens koordinater och lagrar dessa i A och B.

Ditt svar lagras i X.

I raden 70 och 80 kontrollerar datorn att du har svarat rätt.

Den här raden repeterar programmet om C är mindre än 6.

Att köra programmet

Bilden till höger visar vad som händer när du kör programmet. Om ditt svar för dessa två tal multiplicerade med varandra är rätt kommer datorn att skriva ut "fiendens skepp förstört". Är ditt svar inte lika med A*B skriver datorn ut "missat".

```
RUN
FI RYMDSKEPPS KOORDINATER
ÄR 17   3 GE ELD
?41
MISSAT
FI RYMDSKEPPS KOORDINATER
ÄR 11   5 GE ELD
?55
FI SKEPP FÖRSTÖRT
FI RYMDSKEPPS KOORDINATER
ÄR 13   6 GE ELD
```

Kommat på rad 45 åstadkom mellanrummet mellan siffrorna

Övningsuppgift

Kan du addera en räknare till i programmet så att datorn kan räkna ut antal träffar och skriva ut resultatet vid spelets slut? Du måste då sätta upp en variabel kallad S och ge den ett startvärde lika med 0, och sedan addera 1 för varje träff.

Slumptalsprogram

```
5  CLS
10 LET X=INT(RND(1)*30+1)
20 LET Y=INT(RND(1)*30+1)
30 PLOT (X,Y)
40 GOTO 10
```

Denna instruktion tar bort programmet från skärmen innan bilden ritas upp.

Slumptal måste passa din dators skärm.

Den här raden får programmet att upprepas ett oändligt antal gånger.

Datorernas kodord för CLS, RND och PLOT kan variera och några kräver en grafisk rad.



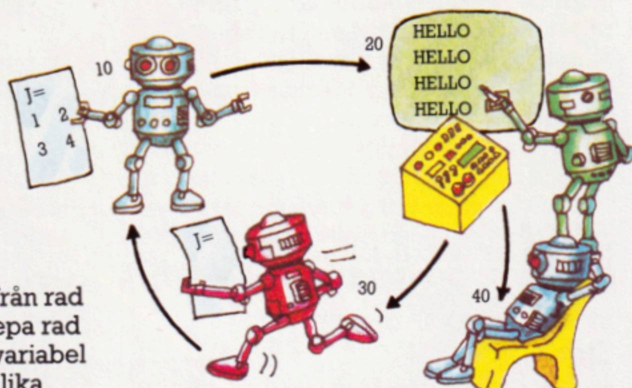
Programmet ovan använder slumptal för att tända punkter på skärmen. Rad 10 och 20 producerar slumptal mellan 1 och 30 och lagrar dem i X och Y. Rad 30 tänder punkten med koordinaterna X,Y. När skärmen fylls upp minskar antalet nytända punkter då en del av de "nya" är redan tända. För att stoppa programmet måste du trycka på BREAK eller ESCAPE, eller motsvarande på din dator.

Att göra en loop

Ofta får datorn göra samma sak ett antal gånger. På sidan 21 kan du se hur delar av programmet kan repeteras med hjälp av GOTO och en variabel som fungerar som en räknare. Ett annat sätt är att upprepa samma rad ett antal gånger med hjälp av FOR...TO och NEXT. Detta kallas att göra en loop.

1 "Hej"-loop

```
10 FOR J=1 TO 6
20 PRINT "HEJ"
30 NEXT            loop
40 END
```



Det här programmet har en loop från rad 10 till 30 vilket får datorn att upprepa rad 20 sex gånger. Bokstaven J är en variabel och rad 10 säger datorn att göra J lika med 1 vid start av programmet, sedan lika med 2 vid andra omgången, lika med 3 vid tredje tills det blir 6. Rad 20 skriver ut ordet hej och rad 30 säger datorn att gå tillbaka och ta nästa värde för J. När J=6 går datorn till rad 40.

2 Enkla beräkningsprogram

```
10 FOR J=1 TO 8            Loop
20 PRINT "2 PLUS 2 ÄR LIKA
   MED 5"
30 NEXT J
40 PRINT
50 PRINT "BARA ETT
   SKÄMT!"
60 END
```

En del datorer har inget utropstecken, så du behöver inte ta med det.

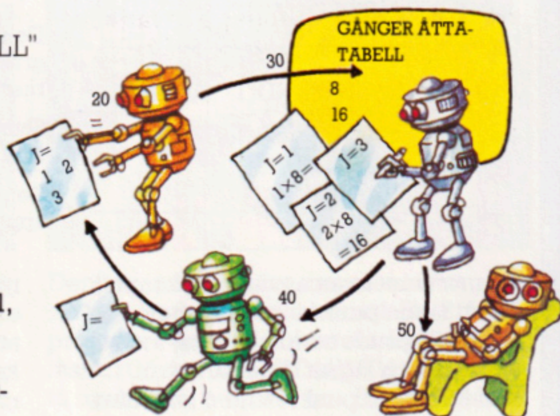
```
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5
2 PLUS 2 ÄR 5

BARA ETT SKÄMT
```

I det här programmet får loopen i raderna 10 till 30 datorn att upprepa rad 20 åtta gånger. Varje gång datorn går genom rad 20 skrivs samma dumma resultat. Efter att ha gjort det åtta gånger fortsätter datorn med resten av programmet. Rad 40 ger bara en tom rad på skärmen.

3 Program för multiplikation

```
10 PRINT "GÅNGER ÅTTA-TABELL"
20 FOR J=1 TO 12
30 PRINT J*8        Loop
40 NEXT J
50 END
```



Den här gången används J för att räkna antal loopar men också som del av J*8. Rad 20 säger datorn att göra J lika med 1, sedan 2 osv upp till 12. Rad 30 tar det aktuella värdet av J, multiplicerar det med 8 och skriver ut svaret. Rad 40 skickar datorn tillbaka till rad 20 och nästa värde för J.

Att göra mönster

FOR...NEXT loopar är mycket användbara för att få fram olika mönster, som det här, med en enkel form som kan upprepas många gånger. Programmet som kan rita mönstret är litet för långt för att skrivas ut helt, men skulle kunna se ut ungefär så här:

```
10 FOR I=1 TO 45
20 Rita en rektangel och ändra dess
   läge litet grand varje gång.
30 NEXT I
40 END
```

Steg

Ibland är det praktiskt att ändra värdet av J med annat än 1. Du kanske vill gå upp med 2 eller ner med 7. För att göra det används ordet STEP. I följande program gör STEP-1 att J minskas med 1 varje gång datorn passerar genom loop i raderna 10 till 40.

Datorprogrammet Glupsk

2-an stoppar loopen (J=2) när det bara finns en kaka kvar.

```
  5 CLS
 10 FOR J=7 TO 2 STEP-1
 20 PRINT "DET FINNS ";J;" KAKOR
    KVAR"
 30 NEXT J
 40 PRINT
 50 PRINT "JAG SPRICKER"
 60 FOR K=1 TO 1000
 70 REM: GÖR INGENTING
 80 NEXT K
 90 PRINT
100 PRINT "B A N G S P L A T T"
```

```
DET FINNS 7 KAKOR KVAR
DET FINNS 6 KAKOR KVAR
DET FINNS 5 KAKOR KVAR
DET FINNS 4 KAKOR KVAR
DET FINNS 3 KAKOR KVAR
DET FINNS 2 KAKOR KVAR

JAG SPRICKER

B A N G S P L A T T
```



Det finns två loopar i det här programmet. En i raderna 10 till 30 får datorn att skriva ut rad 20 sex gånger. För varje gång reduceras J med ett och värdet skrivs ut i rad 20. I loopen i raderna 60 till 80 gör datorn ingenting. Den bara går genom alla värden för K, från 1 till 1000 och det får den att stanna upp ett tag. Rader som börjar med REM (förkortning för REMARK, som betyder kommentar) ger inte instruktionen till datorn. Det finns där bara för att påminna dig om vad programmet gör.

Övningsuppgift

1. Kan du ändra programmet här bredvid och få det att visa "1*8=" tillsammans med svaret?

2. Kan du skriva ett program för "gånger N-tabell", dvs ett program som räknar ut en tabell för varje tal du matar in i datorn? Först måste du få datorn att fråga efter ett tal, N. Sedan får du göra en loop för att räkna och skriva ut resultat. Om du vill så kan du skriva några nya rader vid slutet av programmet som gör att datorn frågar om du vill ha en ny tabell för ett annat tal. Vill du det skall programmet repetera sig självt.

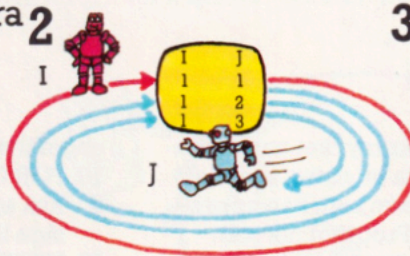
Tricks med loopar

Här får du några fler program med loopar. Nedanför kan du se hur du kan använda loopar för att upprepa flera saker samtidigt.

1 Loopar inuti varandra

```
 5  PRINT "I","J"
10  FOR I=1 TO 3
20  FOR J=1 TO 3
30  PRINT I, J      J loop
40  NEXT J
50  NEXT I     I loop
60  END
```

2



3

```
I   J
1   1
1   2
1   3
2   1
2   2
2   3
3   1
3   2
3   3
```

Det här programmet har både en I- och en J-loop. J-loopen är inbakad inuti I-loopen och för varje gång I-loopen körs upprepas J-loopen tre gånger och J-värdet skrivs ut. Bilden ovan visar resultatet av programmet. Kommatecknen gör att siffrorna ligger en bit ifrån varandra.

Datorklocka

```
 5  CLS
10  LET M=0
20  LET S=0
30  FOR M=0 TO 59
40  FOR S=0 TO 59
50  PRINT M;":";S
60  CLS      sekund-loop
70  NEXT S
80  NEXT M   minut-loop
90  END
```

```
0:45
```

För att få rätt hastighet på klockan kan du använda den här fördröjningsloopen:

```
54 FOR Z=1 TO 100
58 NEXT Z
```



Fel i loopar

```
10  FOR I=1 TO 4
20  FOR J=1 TO 4
30  PRINT I
40  PRINT J
50  NEXT I
60  NEXT J
```

Delar av de inbakade looparna måste vara inuti varandra.



Inuti en dator finns en klocka som bestämmer rytmen för datorns arbete. Denna klocka ger mellan en och fyra miljoner pulser per sekund. Programmet får datorn att uppträda som ett digitalur. Det har inbakade loopar, en för att räkna sekunder och en för minuter.

Sekund-loopen utförs 59 gånger för varje minut-loop. Skulle du prova detta program på en dator kan det hända att klockan kommer att gå för fort. Du får ändra på det genom att lägga in en extra fördröjningsloop i programmet och välja ett passande värde så att den går med samma takt som en vanlig klocka.

Slumptalstester

```
 10  FOR I=1 TO 1000
 20  LET R=INT(RND(1)*6+1)
 30  IF R=1 THEN LET A=A+1
 40  IF R=2 THEN LET B=B+1
 50  IF R=3 THEN LET C=C+1
 60  IF R=4 THEN LET D=D+1
 70  IF R=5 THEN LET E=E+1
 80  IF R=6 THEN LET F=F+1
 90  NEXT I
100  PRINT "FÄRDIGT"
110  PRINT A, B, C
120  PRINT D, E, F
130  END
```



Det här programmet tar tar mycket lång tid att utföra. Du kan korta av det genom att ändra talet i rad 10 till 500 eller 250.

```
RUN
FÄRDIGT
162        168        167
160        187        156
```

Programmet visar om RND verkligen fungerar. Loop i rad 10 till 90 får datorn att ett tusen gånger ta fram ett slumptal som ligger mellan 1 och 6. Datorn räknar sedan hur många gånger varje tal kommer upp och lagrar resultat i variablerna A till F för att sedan skriva ut resultaten (*).

* I vissa datorer, t ex ZX81, behövs några extra rader i början av programmet för att nollställa alla variabler.

Program som upprepar ett mönster

Programmet nedan använder flera loopar för att placera ett litet mönster över hela skärmen. Det ser ganska komplicerat ut, men om du läser genom det och tar reda på vad varje rad gör, kommer du snart underfund med hur det fungerar. Var på skärmen mönstret hamnar bestäms av slumptal och blir olika för varje gång du kör programmet.

På datorer som har högupplösningsgrafik får du använda större slumptal.



```
5 CLS
10 LET A=INT(RND(1)*6+1
20 LET B=INT(RND(1)*7+1
30 LET C=INT(RND(1)*6+1
40 LET D=INT(RND(1)*4+1
50 INPUT "HUR MÅNGA PUNKTER
   HORISONTELLT";W
60 INPUT "HUR MÅNGA VERTIKALT";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6
90 PLOT (J+A,I+B)
100 PLOT (J+A,I+C)
110 PLOT (J+C,I+D)
120 PLOT (J+B,I+D)
130 NEXT J
140 NEXT I
150 END
```

J loop

I loop

Dessa rader väljer slumptal för vårt mönster och lagrar dem i A, B, C och D.

Rad 50 och 60 frågar efter skärmens bredd (W) och höjd (V).

I-loopen räknar hur många gånger mönstret upprepas längs den vertikala axeln. För varje gång ökas I med skärmens höjd (V) dividerad med 6, så mönstret upprepas 6 gånger.

Varje gång loopar upprepas får raderna 90 till 120 datorn att tända fyra punkter med hjälp av aktuella värden för I och J plus slumptalen.

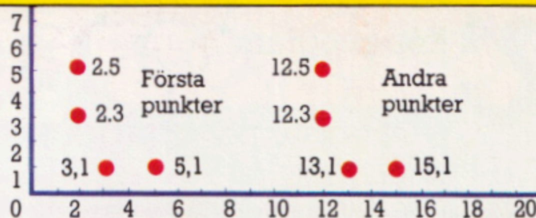
J-loopen räknar hur många gånger mönstret upprepas längs den horisontella axeln. Det fungerar på samma sätt som I-loopen.

1 Hur fungerar det



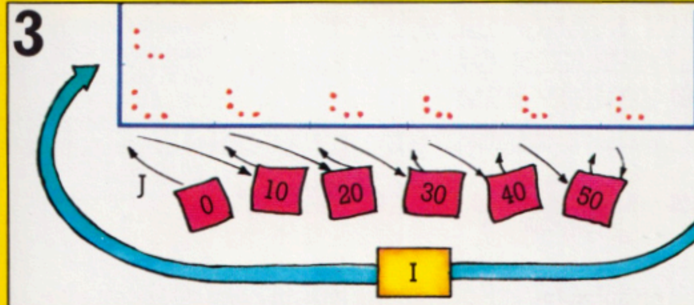
Föreställ dig att datorn har valt 2, 5, 3 och 1 som slumptal och att skärmens bredd och höjd är 60.

2



Då programmet utförs för första gången är I och J lika med 0, så datorn tänder de första punkterna genom att använda endast slumptal. Rad 130 skickar dem tillbaka för att få fram nästa värde för J, vilket blir J+60/6, dvs 10. Nu tänds nya punkter med koordinaterna lika med slumptal plus 10 för J. På så sätt sprids punkterna över hela skärmen.

3



Om du provar programmet och det inte fungerar, försök igen med mindre värden för V och W.

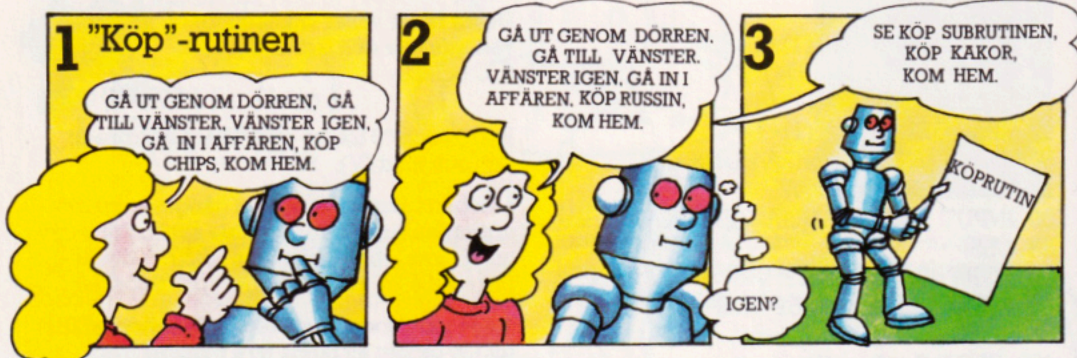


J-loopen upprepas 6 gånger och varje gång adderas 10 till J så att fler punkter tänds längs den vertikala axeln. Sedan går datorn tillbaka för att få fram nästa värde för I, vilket är 10. J sätts till 0 och datorn tänder nästa rad med punkter genom att använda 10 för I och öka J med 10 varje gång som tidigare.

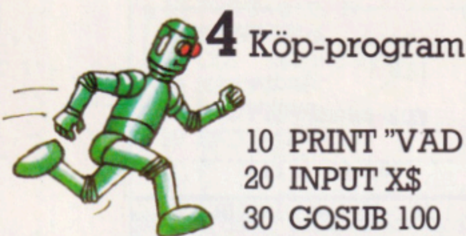
Övningsuppgift – Kan du skriva ett program som upprepar rymdfiendens figur över hela skärmen? Det finns några exempel på sidan 45.

Subrutiner

En subrutin är ett litet program inuti ett program. Den utför en speciell funktion, t ex adderar tal, och du kan skicka datorn dit varje gång du vill ha denna funktion utförd. Detta gör att du inte behöver skriva samma programrader flera gånger, programmet blir kortare och enklare att läsa och förstå.



Tänk dig att du har en robot som kan programmeras för att lyda dina order. Om du skulle vilja ha något från affären kan du ge roboten exakta instruktioner hur den skulle hitta dit. Varje gång roboten ska gå och handla måste du ge den samma instruktioner. Det blir mycket enklare att ge roboten köprutinen och referera till den vid behov.



4 Köp-program

```
10 PRINT "VAD VILL DU HA FRÅN AFFÄREN"
20 INPUT X$
30 GOSUB 100
40 PRINT "NÅGONTING ANNAT"
50 INPUT M$
60 IF M$="JA" THEN GOTO 10
70 STOP

100 REM: KÖPSUBRUTIN
110 PRINT "GÅ UT, SVÄNG TILL VÄNSTER"
120 PRINT "VÄNSTER IGEN, GÅ IN I AFFÄREN"
130 PRINT "KÖP ";X$ KOM HEM"
140 RETURN
```

Rad 30 skickar datorn till första subrutinraden.

STOP behövs vid slutet av huvudprogrammet för att hindra datorn från att gå in i subrutinen.

Det är ganska praktiskt att sätta en etikett på subrutinen med hjälp av en REM-rad, så du vet vad den gör.

Det här sänder datorn tillbaka till rad 40 efter subrutinen.



För att få datorn att hoppa till en subrutin, används ordet GOSUB med RETURN i slutet av subrutinen. GOSUB måste följas av numret på första subrutinraden. RETURN behöver inget radnummer. Datorn går automatiskt tillbaka till raden efter den från vilken hoppet till subrutinen utfördes. Du kan skicka datorn till en subrutin så många gånger som du vill i ett program.

GOSUB-program

En subrutin är mycket användbar för att utföra någonting som du vill ha gjort flera gånger i olika delar av programmet. Här kan du se några fler program med subrutin.

Beräkningsprogram

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT "A DIVIDERAT MED B=";A/B
90 GOTO 50
250 REM: SUBRUTIN FÖR STOP
260 IF A=0 AND B=0 THEN STOP
270 RETURN
```

Denna subrutin ger dig ett uthopp från programmet. Vill du sluta dividera får du mata in 0 i rad 50 och 60. Det här programmet behöver inte STOP före subrutinen därför att rad 90 skickar det tillbaka.

Omräkningsprogram

```
100 INPUT "AVSTÅND";M
110 INPUT "TID";T
120 GOSUB 200
130 PRINT "GENOMSNITTLIG
    HASTIGHET VAR"
140 PRINT M/T;"M/TIM OCH";K/T;"KM/TIM"
150 STOP
200 REM: SUBRUTIN FÖR ATT RÄKNA OM MILES
210 LET K=M*1.609
220 RETURN
```

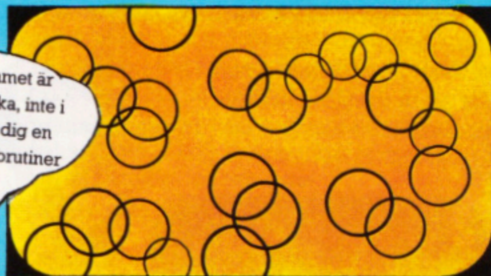
Den här subrutinen räknar om en engelsk mil till kilometer. Samma subrutin kan användas i flera olika program. Men kontrollera att du använder samma variabelnamn.

Program som ritar cirklar

```
1 CIRKELNS CENTRUM = X,Y
2 CIRKELNS RADIE = R
3 FÄRG = X
4 GOSUB 10
5 GOTO 1
10 REM: SUBRUTIN SOM RITAR CIRKLAR
11 RITA EN CIRKEL MED CENTRUM
   X,Y; RADIE R OCH FÄRG X.
12 RETURN
```

Det här programmet är skrivet på svenska, inte i BASIC, för att ge dig en känsla för hur subrutiner används.





Subrutiner är användbara i grafikprogram som det här för att rita diagram med hjälp av värden som räknats ut i huvudprogrammet. Med det här programmet skulle du kunna rita många olika cirklar genom att ge datorn olika värden i rad 1 till 5.

Frågeprogram

```
5 LET C=0
10 PRINT "NÄR UPPFANNS DESSA SAKER?"
20 READ C$,F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10

100 REM: SVARSSUBRUTIN
110 IF ABS(A-F)<10 THEN PRINT "OK"
120 IF ABS(A-F)>10 THEN PRINT "NEJ"
130 PRINT "FÖRSÖK EN GÅNG TILL"
140 RETURN

200 DATA TELEFON, 1876, TRYCKPRESS, 1450, CYKEL, 1791
```

I rad 20 hämtar datorn första ordet och första talet från datasatsen och placerar dem i C$, respektive F.

Här skrivs innehållet i C$ ut.

Räknaren C stoppar programmet efter tre gånger eftersom det bara finns tre poster i datasatsen.

Det här är subrutinen.

Varje gång programmet utförs ersätts innehållet i C$ och F med nästa post i datasatsen.

Det här programmet använder en subrutin för att kontrollera svaren. Korrekta svar finns i F och de inmatade svaren i A. Med hjälp av rad 100 och 110 jämför datorn A med F. ABS betyder absolut och får datorn att räkna ut skillnaden mellan A och F (resultatet blir alltid ett positivt tal). Är skillnaden mindre än 10 skriver datorn OK, är den större än 10 blir det NEJ.

Arbeta med ord

De flesta datorer kan testa ord som finns lagrade i variablerna och göra olika saker med dem. Datorn kan t ex pröva om innehållet i en variabel är ett speciellt ord eller tecken. Detta är ganska praktiskt för att kontrollera orden som matas in av den som använder programmet. Datorer kan också flytta om bokstäver och ord i variablerna och lägga ihop dem med innehållet i andra variabler. Här nedan kan du se hur du kan göra det i BASIC.

1

```
10 A$="JAG ÄR KORKAD"
20 B$="BARA DUMMA TYCKER"
30 C$=B$+" "+A$
RUN
BARA DUMMA TYCKER JAG ÄR KORKAD
```

ZX81 behöver ordet LET de flesta andra klarar sig utan.

Du kan lägga ihop innehållet i två variabler så här. Utrymme behövs mellan citationstecknen för att få avstånd mellan orden.

2

```
PRINT LEFT$(B$,4)
ONLY
PRINT LEFT$(B$, 4)+" "+A$.
BARA JAG ÄR KORKAD
```

Du kan också lägga ihop delar av variablernas innehåll, som ovan. LEFT$(B$,4) betyder "ta de fyra första bokstäverna från vänster i B$."

A$
B$
JAG ÄR KORKAD
BARA DUMMA TYCKER

3

```
PRINT RIGHT$(A$,6)
KORKAD
```

För att få datorn att ta bokstäver från höger skriver du RIGHT$ tillsammans med variabelnamnet och antalet bokstäver du vill ha utskrivna.

4

```
PRINT MID$(B$,6,5)
DUMMA
```

Det här säger datorn att ta bokstäverna i mitten. Första talet anger var den skall börja och det andra hur många bokstäver som skall hämtas.

När du räknar bokstäver skall du ta med även mellanslag och interpunktionstecken.

5

```
10 K$="DING DONG"
20 PRINT LEN(K$)
RUN
10
```

Du kan också ta reda på hur lång teckensträngen är, dvs hur många bokstäver, mellanslag och symboler den innehåller. För det används LEN, förkortning för längd, length, på engelska.

BARA
DUMMA
TYCKER

En information för Sinclair-användare.

```
PRINT A$(8 TO 13)
KORKAD
PRINT B$(15 TO 17)
KER
```

Det betyder: tag bokstäverna från nummer 8 t o m 13.

Sinclair-datorer använder inte LEFT$, RIGHT$ och MID$, men du kan få datorn att ta vilka bokstäver som helst med hjälp av instruktioner ovan.



IF A$="DATOR BOK" vad blir LEFT(A$,5)? RIGHT$(A$,6)? MID$(A$,4,7)?

Kodprogram

Det här programmet kodar automatiskt dina meddelanden. Likadana, men mycket mera komplicerade, program används av underrättelsetjänster för att skriva eller knäcka koder.
Det bästa sättet att förstå hur programmet fungerar är att skriva ett hemligt meddelande på ett papper och sedan gå igenom programraderna och för hand göra datorns arbete med meddelandet.

```
5 LET C$"=" "
7 LET D$"=" "
10 PRINT "SKRIV IN ETT KORT MEDDELANDE"
20 INPUT M$
30 PRINT "SKRIV NU IN ETT HEMLIGT
   NUMMER MELLAN 2 OCH ";LEN(M$)-1
40 INPUT N
50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$+MID$(M$,I,1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J,1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "DET KODADE MEDDELAN-
    DET ÄR"
160 PRINT M$
170 END
```

Rad 5 och 7: Ger strängvariabeln värdet blank.

Rad 30: Det här betyder längden av ditt meddelande minus 1.

Rad 50: N (ditt hemliga nummer) bokstäver från höger av M$.

Rad 60: Längden av M$ minus N antal bokstäver från vänster av M$ (dvs resten).

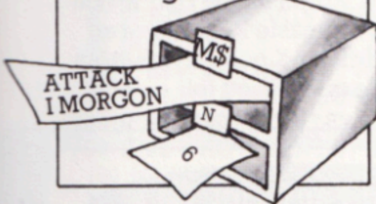
Rad 70: Ersätter bokstäverna i M$ med A$+B$.

Rad 80: Varannan bokstav från 1 till den sista i ditt meddelande, dvs 1, 3, 5 osv. Varje gång I-loopen upprepas tar rad 90 en bokstav från position I av M$ och lägger den i C$.

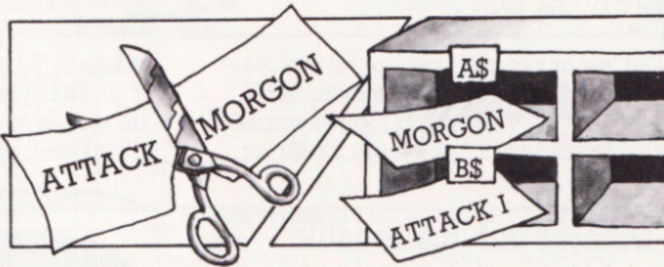
Rad 110: Varannan bokstav från 2 till den sista i ditt meddelande, dvs 2, 4, 6 osv. Det fungerar på samma sätt som loopen ovan.

Rad 140: Ersätter igen bokstäverna i M$.

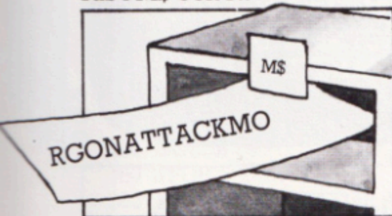
Så fungerar det



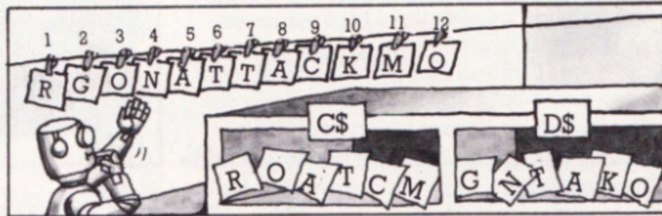
Antag att ditt hemliga meddelande är "attack i morgon" och att den hemliga siffran är 6. Dessa lagras i M$ och N.



I rad 50 och 60 använder datorn den hemliga siffran för att dela meddelandet. I rad 50 tas sex bokstäver från höger och lagras i A$. I rad 60 tas resten och lagras i B$.



I rad 70 adderas A$ och B$. Då hamnar de sista bokstäverna i början av meddelandet.

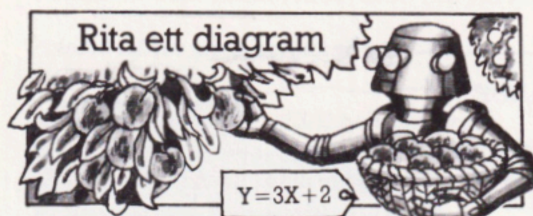


Varje gång loop I upprepas lagras en bokstav med ett udda nummer i C$ (t ex o, r, m osv). Varje gång J-loopen upprepas lagras en bokstav med ett jämnt nummer i D$ (t ex n, g, i osv). Sedan adderas C$ och D$ och du får ditt kodade meddelande.

Diagram och symboler

Du kan programmera en dator så att den presenterar information på flera olika sätt, t ex med ord, tal, bilder eller diagram. Det kan vara mycket lättare att förstå komplicerade sammanhang om du illustrerar med hjälp av symboler eller grafik.

Rita ett diagram
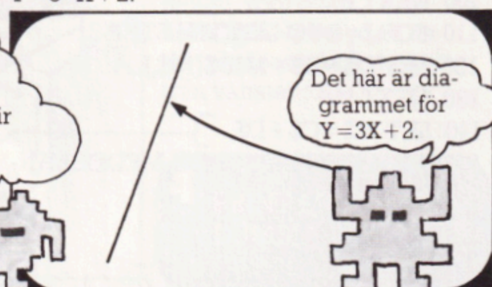




Tänk dig ett persikoträd som bär mer och mer frukt för varje år i takt med trädets åldrande. Det kan beskrivas med hjälp av en ekvation, t ex Y=3X+2 (Y är mängden frukt och X åldern), men den är nog ganska svår att fatta. Ett diagram skulle vara mycket enklare.

Med en dator är det ganska lätt att rita ett diagram som visar hur Y ändras i förhållande till X. För att kunna rita diagrammet måste du ha ett värde för Y för varje värde av X. Med programmet görs detta enkelt med följande instruktion LET Y=3*X+2.

Instruktioner för CLS och PLOT kan vara olika på olika datorer.

```
5 CLS
10 FOR X=1 TO 14
20 LET Y=3*X+2
30 PLOT (X,Y)
40 NEXT X
50 END
```

Max värdet för X blir 14 och max värdet för Y blir 3*14+2.



Det här är programmet för att rita diagrammet. Loopen sätter värden för X från 1 till 14. Varje gång loopen upprepas används det nya värdet i X för att räkna ut Y i rad 20 och rad 30 ritar X och Y på skärmen. Du måste kontrollera att de största värden för X och Y ryms inom din skärm, annars får du ett fel.

Datorer och matematik

I ett matematiskt uttryck med flera delar, som t ex 3*X+2, multiplicerar eller dividerar datorn först för att sedan addera eller subtrahera. Det betyder att vi skulle få samma svar för de här båda exemplen.

```
PRINT 4*6+8        PRINT 8+4*6
32                 32
```

Om du vill att datorn ska göra beräkningar i en annan ordning får du använda parenteser:

```
PRINT (8+4)*6
72
```

Den här gången adderar datorn 8 och 4 och multiplicerar sedan med 6.

Övningsuppgift

TÄNK UT ETT TAL
FÖRDUBBLA DET, ADDERA 4
DIVIDERA DET MED 2, ADDERA 7
MULTIPLICERA MED 8, TA BORT 12
DIVIDERA MED 4 OCH TA BORT 11
HUR MYCKET BLEV DET?
TALET DU TÄNKTE PÅ FÖRST VAR...

Kan du skriva ett program för att få datorn att genomföra det här kända tricket? (För att få fram det talet som du först tänkte på, tar du bort 4 från resultatet och dividerar sedan med 2.)

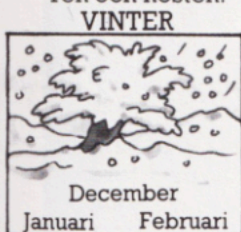
Födelsedagsprogram

Det här programmet använder ett annat sätt att visa information. Det använder symboler för att jämföra antalet människor som var födda under olika årstider. Du skulle också kunna använda programmet för att få fram t ex hur ofta en viss fågel visar sig under olika årstider, eller hur många gånger olika fotbollslag har vunnit. Det är en bra idé att göra en programplan innan du skriver ett långt program som det här.

Programplan

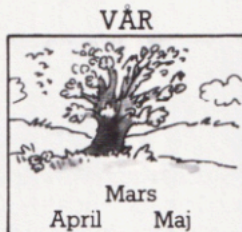
Mål: att jämföra antal människor med födelsedagar under vintern, våren, sommaren och hösten.

VINTER



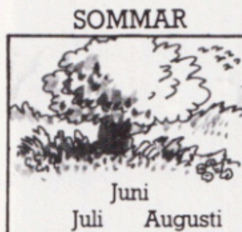
December
Januari Februari

VÅR



Mars
April Maj

SOMMAR



Juni
Juli Augusti

HÖST



September
Oktober November

1. Ge datorn information (data), dvs årstider när de, säg 20 testade personer, föddes.
2. Lagra informationen i datorn.
3. Presentera informationen på skärmen.

En provkörning

```
RUN
TOTALT PÅ VINTERN************
TOTALT PÅ VÅREN*****
TOTALT PÅ SOMMAREN******
TOTALT PÅ HÖSTEN******
```

Programmet

```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
10 FOR I=1 TO 20
20 PRINT "PERSON ";I;" FÖDDES PÅ"
30 PRINT "VINTERN, VÅREN, SOMMAREN
   ELLER HÖSTEN"
40 PRINT "TYPE VI, VÅ, SO ELLER HÖ"
50 INPUT B$
60 IF B$="VI" THEN LET A=A+1
70 IF B$="VÅ" THEN LET B=B+1
80 IF B$="SO" THEN LET C=C+1
90 IF B$="HÖ" THEN LET D=D+1
100 NEXT I
110 PRINT "TOTALT PÅ VINTERN";
115 LET N=A
120 GOSUB 200
130 PRINT "TOTALT PÅ VÅREN";
135 LET N=B
140 GOSUB 200
150 PRINT "TOTALT PÅ SOMMAREN";
155 LET N=C
160 GOSUB 200
170 PRINT "TOTALT PÅ HÖSTEN";
175 LET N=D
180 GOSUB 200
190 STOP
200 REM: SUBRUTIN FÖR ATT RITA STJÄRNOR
210 IF N=0 THEN GOTO 250
220 FOR I=0 TO N
230 PRINT "*";
240 NEXT I
250 PRINT
260 RETURN
```

Rad 5–8: Tomma variabler för totaler för varje årsdel.

Rad 10: Loop för en fråga per person i gruppen.

Raderna 60 till 100 kontrollerar svaret i B$ och adderar ett till variabeln för rätt årstid.

Rad 100: Skickar datorn tillbaka för att upprepa frågan.

Rad 120: Subrutinen som får datorn att skriva ut antal stjärnor lika med talet i varje variabel.

Rad 135–155: Genom att lagra totalen i N varje gång kan samma rutin användas för alla årstider.

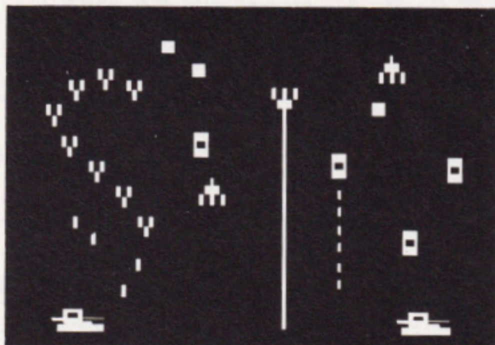
Rad 170: Får datorn att stanna på samma rad när stjärnorna skrivs ut.

Rad 210 används om ingen föddes under någon årstid.

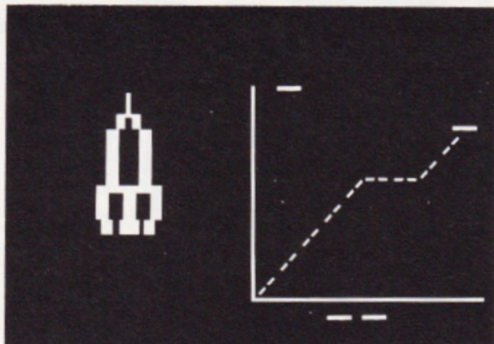
Rad 220: Huvudprogrammet sätter N lika med totaler för A, B, C och D. Loopen får datorn att upprepa rad 230 N gånger.

Mera grafik

De här två sidorna visar dig hur du kan använda PLOT och UNPLOT för att få rörliga bilder på skärmen. Det kallas grafik och är mycket användbart för spel eller program som förklarar t ex gravitationsprincipen.



Bilden i både en video och ett arkad-spel styrs av en liten dator. Datorn är programmerad endast för spel, och programmen är skrivna inte i BASIC utan i datorns speciella kod.



En generell dator, programmerad i BASIC, ger långsammare, enklare bilder. Den klarar inte av alla skärminstruktioner tillräckligt snabbt för att det skall bli rörliga bilder.

PLOT/UNPLOT-program

1

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```



Det här korta programmet får en ljuspunkt att röra sig över skärmen. Kom ihåg att PLOT och UNPLOT instruktionerna kan vara olika i olika datorer.



Skulle punkten nå skärmens kant kan det hända att programmet stannar och du får ett felmeddelande om att du har hamnat utanför skärmen.

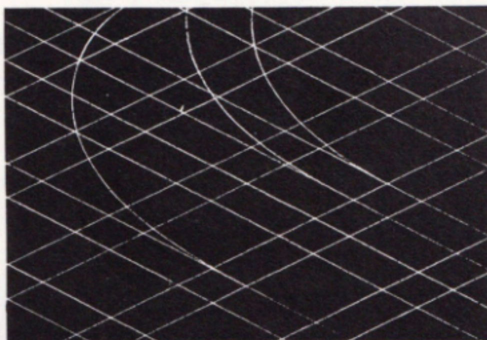
3

Racket- och bollspel använder program som ovan för att flytta bollen över skärmen. Det finns enkla regler hur du kan få bollen att fortsätta även när den når skärmens kant.

När bollen når skärmens övre kant subtraheras värdet som skulle adderats till Y. På samma sätt, när bollen når en av kanterna på sidan, subtraheras värdet från X.

Linjemönsterprogram

Detta program ritar en linje över skärmen och när linjen når en av kanterna skickas den tillbaka åt ett annat håll. Programmet använder inte UNPLOT, så linjerna ligger kvar på skärmen. Bilden till höger visar vad som händer om du kör det här programmet. Rad 100 får programmet att rita 10 000 punkter. För att göra programmet kortare kan du minska det, eller använda BREAK för att stoppa det.



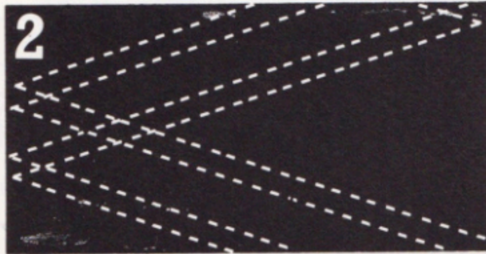
```
10 REM: INITIERA GRAFIKMOD OM NÖDVÄNDIGT"
20 PRINT "HUR MÅNGA PUNKTER
   VERTIKALT?";
30 INPUT H
40 PRINT "OCH HORISONTELLT";
50 INPUT V
55 CLS
60 LET X=H/2
70 LET Y=V/2
80 LET S=1
90 LET T=1
100 FOR I=1 TO 10000
110 LET S=S+(INT(RND(1)*10+1)-5)/50
120 LET X=X+S
130 LET Y=Y+T
140 IF X<5 THEN LET S=-S
150 IF X>H-5 THEN LET S=-S
160 IF Y<5 THEN LET T=-T
170 IF Y>V-5 THEN LET T=-T
180 GOSUB 300
190 NEXT I
200 STOP
300 REM:RITA LINJEN
310 PLOT (X,Y)
320 RETURN
```

- Rad 20–50: I rad 20 till 50 frågas efter skärmens höjd och bredd. Semikolon gör att ditt svar hamnar på samma rad som frågan.
- Rad 60–70: Det här får X och Y att börja i mitten av skärmen.
- Rad 80–90: S och T är värden som skall adderas till X och Y för att få linjen att röra på sig.
- Rad 100: Loopen från rad 100 till 190 upprepas 10 000 gånger. Varje gång ändras X och Y en liten bit.
- Rad 110: Den här formeln producerar värdet som skall adderas till X. Formeln ger ett litet värde som ändras för varje gång loopen upprepas.
- Rad 140–170: Dessa rader kontrollerar att skärmens gränser inte överskrids. När X eller Y hamnar 5 punkter från kanten byts tecknet på S och T.
- Rad 180: Skickar datorn till ritsubrutinen.
- Rad 310: Ritar punkten med aktuella värden för X och Y.

Experiment



Rad 110 adderar ett litet slumptal till X som får linjen att gå i zig-zag över skärmen. Har du en dator kan du försöka ta bort den raden. Linjerna på skärmen blir då parallella.

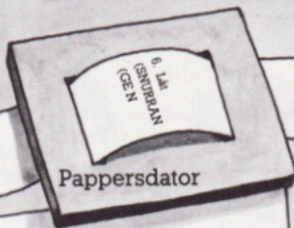


Ändra värdena i rad 80 och 90 till 5 eller 10 t ex (eller ännu större i en dator med högupplösningsgrafik). Då får du en bruten linje.

Roliga poesiprogram

På de följande sidorna ser du hur du kan skriva program som kan komponera många "dikter".

Program för pappersdator



Snurra

1 A=0 och B=0
2 Addera 1 till A
3 Om A=6 koppla till rad 10
4 Skriv dataraden A
5 Addera 1 till B
6 Låt snurran ge N
7 Skriv dataord från rad B kolumn N
8 Om B=3 eller 5 hoppa till rad 5
9 Hoppa till rad 2
10 Stopp

Datarader

DET VAR EN UNG MAN FRÅN
SOM
HANS
MEN DAGEN DÄRPÅ
OCH HAN DRACK ALDRIG MER

Dataord

BERLIN	TURIN	WIEN	ABERDEEN
TYCKTE	DRÖMDE	FICK FÖR SIG	TRODDE
FRU	FOT	KATT	KROPP
VAR GREDELIN	STANK MARGARIN	VAR EN PINGVIN	SATT I GARDIN
VAR HAN VISSEN	VAR HAN SKAMSEN	VAR HAN LIKBLEK	VAR HAN BAKIS
SOM FÅ	OCH GRÅ	FÖR TVÅ	OCH SÅ
SOM ETT SVIN	NÅGON GIN	TERPENTIN	ÖL OCH VIN

Det var en ung man från Berlin
Som trodde hans katt var gredelin
Men dagen därpå
Var han vissen för två
Och han drack aldrig mer öl och vin

Det här är ett program för en pappersdator. Det påminner litet om ett BASIC-program, men skulle inte fungera på en riktig dator. Ord och fraser för dikten är lagrade på papperslappar och programmet talar om för dig vilka som skall väljas. Snurran är en slumptalsgenerator som ger slumptal mellan ett och fyra.

1 Programmet översatt till BASIC

Det kommer ännu inte att fungera på en dator.

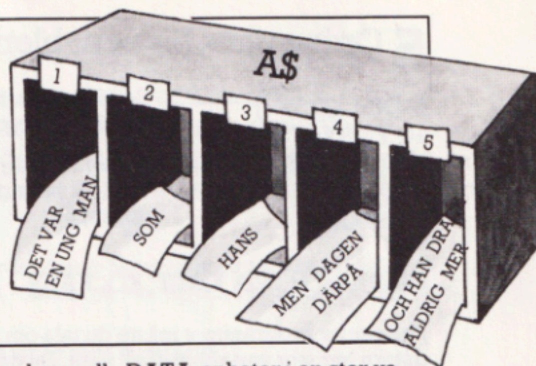
```
10 LET A=0
20 LET B=0
30 LET A=A+1
40 IF A=6 THEN STOP
50 Skriv dataraden A
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 Skriv dataord från rad B
   kolumn N
90 IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

- Rad 10–20: Den här raden sätter upp tomma variabler.
- Rad 30–40: Rad 30 och 40 håller reda på hur många datarader datorn har valt.
- Rad 50: Rad 50 och 80 är ännu inte skrivna i BASIC.
- Rad 60: Rad 60 håller reda på antalet dataord.
- Rad 70: Ger ett slumptal mellan ett och fyra.
- Rad 90–100: Rad 90 och 100 skickar datorn tillbaka för att välja en annan datarad.

Större delen av programmet är enkelt att översätta till BASIC, men rad 50 och 80 är svårare. Datorn måste ha ett sätt att lagra och hämta datarader och ord som behövs för varje rad i dikten.

2 Att mata datorn med data

```
50 READ A$
:
180 DATA DET VAR EN UNG MAN FRÅN,
    SOM, HANS
190 DATA MEN DAGEN DÄRPÅ, OCH HAN
    DRACK ALDRIG MER
```



För att ge datorn datarader och ord kan du använda READ....DATA. Varje gång som datorn utför READ-instruktionen hämtar den en ny enhet ur DATA-raden och lagrar den i variabeln.

Du kan lagra alla DATA-enheter i en stor variabel A$. En variabel som innehåller mer än en dataenhet kallas fält. Du refererar till en enhet med hjälp av ett nummer, t ex READ A$(3), vilket här ger HANS.*

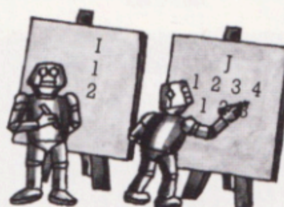
3



En variabel kan också innehålla flera rader av data och du kan lagra alla dataord i en variabel som den här. Den kallas tvådimensionellt fält. Här refereras till varje enhet med hjälp av rad- och kolumnnummer. Så att READ B$(4,2) skulle ge STANK MARGARIN och READ B$(6,3) FÖR TVÅ. Du kan lagra tal i fält också, då använder du en talvariabel, t ex N(5,7).

4 Lagring av data i variablerna

```
10 FOR I=1 TO7   ——— I är radnumret
20 FOR J=1 TO4   ——— J är kolumnnumret
30 READ B$(I,J)
40 NEXT J
50 NEXT I
60 DATA BERLIN, TURIN, WIEN, ABERDEEN
70 DATA TYCKE, DRÖMDE, FICK FÖR SIG, TRODDE
80 DATA FRU, FOT, KATT, KROPP
90 DATA VAR GREDELIN, STANK MARGARIN, VAR EN PINGVIN, SATT I GARDIN
100 DATA VAR HAN VISSEN, VAR HAN SKAMSEN, VAR HAN LIKBLEK, VAR HAN BAKIS
110 DATA SOM FÅ, OCH GRÅ, FÖR TVÅ, OCH SÅ
120 DATA SOM ETT SVIN, NÅGON GIN, TERPENTIN, ÖL OCH VIN
```





För att skriva in dataenheter i variablerna måste du kunna nå de tal som finns inom paranteser efter READ. Det kan du göra med loopar. En loop inuti en annan loop behövs för B$.

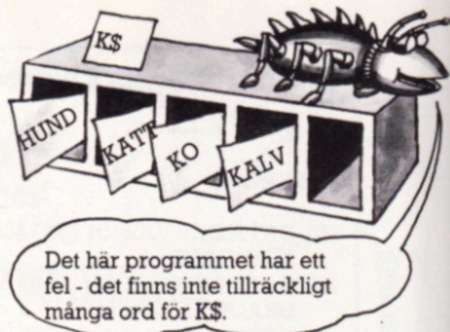
I programmet ovan ger I-loopen radnummer och J-loppen kolumnnummer. För varje I-lopp görs fyra J-loopar, en för varje kolumn i raden.

* Sinclair-datorer arbetar på ett annat sätt med variablerna och detta program kan därför ej användas på en Sinclair.

5 Ordna plats för variablerna

```
5 DIM K$(5)         ]——— Detta är variabelns storlek,
10 FOR I=1 TO 5           dvs 5 ord i en rad.
20 READ K$(I)       ]——— Denna rad stoppar in data-
30 NEXT I                 ord i K$ varje gång loopen
40 STOP                   upprepas.

60 DATA HUND, KATT, KO, KALV
```



I början av programmet måste du tala om för datorn hur stor variabeln skall vara. Du gör detta med ordet DIM, som följs av variabelns namn och antal dataord t ex DIMK$(5).

För ett tvådimensionellt fält får du ge datorn antal rader och kolumner i variabeln, t ex DIMK$(5,3). Du måste alltid ha rätt värde för antal variabler, annars blir det fel.

6 Utskrift av data

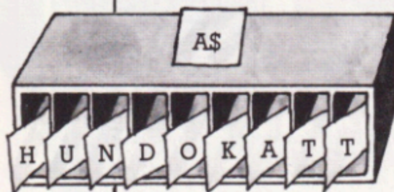
```
200 LET A=0
210 LET B=0
220 LET A=A+1          ]——— A håller reda på hur många gånger denna
230 IF A=6 THEN STOP         sektion av programmet upprepas.
240 PRINT A$(A)
250 LET B=B+1          ]——— B håller reda på antalet rader med data och
260 LET N=INT(RND(1)*4+1)    ser till att korrekt rad används.
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250 ]——— Rad 280 och 290 får datorn att skriva ut orden
290 IF B=5 THEN GOTO 250 ]    från föregående rad innan den aktuella raden
                              skrivs ut.
300 GOTO 220           ]——— Det här skickar datorn tillbaka för utskrift av
310 END                      nästa datarad.
```

Ett sådant program behövs för att datorn skall kunna skriva ut data i rätt ordning. Den här sektionen av programmet upprepas 5 gånger. Varje gång skriver datorn ut dataradens nummer A och några ord från rad B. Vilka ord som skall skrivas ut beror på slumptalet i N.

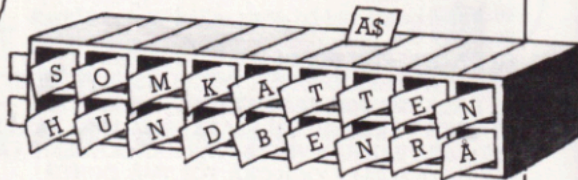
Sinclairdatorer och variabler

Det här programmet fungerar inte på Sinclairdatorer, eftersom de arbetar med strängar på ett annat sätt.

För tvådimensionella fält får du tala om för datorn, utöver det ovannämnda, även radnummer. Till exempel A$(2,1 TO 4) ger HUND.







För att få en Sinclairdator att plocka ett speciellt ord från en variabel måste du uppge numret för det första och sista tecknet i ordet. Det är samma system som Sinclair använder för LEFT$, RIGHT$ osv. (Se sid 32.)

I början av programmet måste du tala om för datorn hur många rader ditt fält skall ha och hur många tecken det kommer att finnas i varje, t ex DIMA$(2,9). Det betyder två rader, varje med 9 tecken. Varje rad i ett fält måste ha lika många tecken.

Ett komplett skämtpoesiprogram

Nu kan du sätta ihop olika delar av programmet och skriva ett komplett poesiprogram. Första delen av programmet (rad 10 till 190) ger datorn data och andra delen (rad 200 till 310) skriver ut en dikt. Varje gång du kör programmet får du en annorlunda version av dikten därför att slumptalet i N får datorn att välja olika ord.

```
10 DIM A$(5)
20 DIM B$(7,4)
30 FOR I=1 TO 7
40 FOR J=1 TO 4
50 READ B$(I,J)
60 NEXT J
70 NEXT I
80 DATA BERLIN, TURIN, WIEN, ABERDEEN
90 DATA TYCKTE, DRÖMDE, FICK FÖR SIG, TRODDE
100 DATA FRU, FOT, KATT, KROPP
110 DATA VAR GREDELIN, STANK MARGARIN, VAR EN PINGVIN, SATT I GARDIN
120 DATA VAR HAN VISSEN, VAR HAN SKAMSEN, VAR HAN LIKBLEK, VAR HAN
    BAKIS
130 DATA SOM FÅ, OCH GRÅ, FÖR TVÅ, OCH SÅ
140 DATA SOM ETT SVIN, NÅGON GIN, TERPENTIN, ÖL OCH VIN
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA DET VAR EN UNG MAN FRÅN, SOM, HANS
190 DATA MEN DAGEN DÄRPÅ, OCH HAN DRACK ALDRIG MER
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Rad 10 och 20 talar om för datorn hur mycket utrymme den skall lämna för variablerna – en rad på 5 för A$ och 7 rader på 4 för B$.

Detta är raden av loop för att mata in data i B$.

Raderna 80 till 140 innehåller alla dataord som skall lagras i B$.

Detta är en loop för att mata in data i A$.

Raderna 180 och 190 innehåller alla dataord som skall lagras i A$.

Med detta skriver datorn ut den rad som är lagrad i A$ vid bokstaven A.

Med detta får datorn order att skriva dataorden lagrade i B$ rad B kolumn N.

Programmet stannar vid rad 230 om A=6, så den når aldrig rad 310, men somliga datorer måste ändå få en END-rad.

Provkörningar

```
DET VAR EN UNG MAN FRÅN
WIEN
SOM
FICK FÖR SIG
HANS
FRU
VAR EN PINGVIN
MEN DAGEN DÄRPÅ
VAR HAN BAKIS
SOM FÅ
OCH HAN DRACK ALDRIG MER
SOM ETT SVIN
```

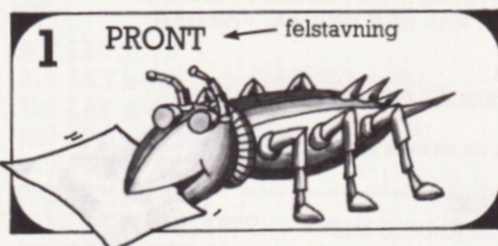
```
DET VAR EN UNG MAN FRÅN
TURIN
SOM
TYCKTE
HANS
FOT
STANK MARGARIN
MEN DAGEN DÄRPÅ
VAR HAN LIKBLEK
OCH GRÅ
OCH HAN DRACK ALDRIG MER
NÅGON GIN
```

Här får du två av 16 384 möjliga olika versioner av dikten. Om du försöker köra det här programmet och alltid får samma version från din dator så titta efter i handboken hur du skall generera olika slumptal. En del datorer ger alltid samma följd av slumptal.

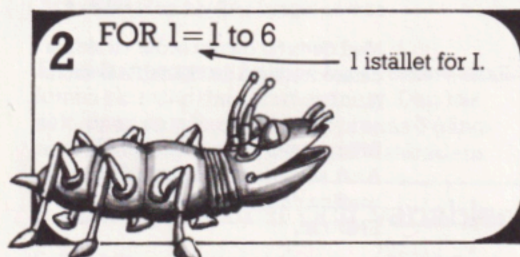
Programmeringstips

På de här två sidorna får du några tips som skall hjälpa dig att skriva egna program och en förteckning över de vanligaste felen samt deras orsaker. Oftast förekommande fel står först, så titta igenom listan om du har fel i ditt program.

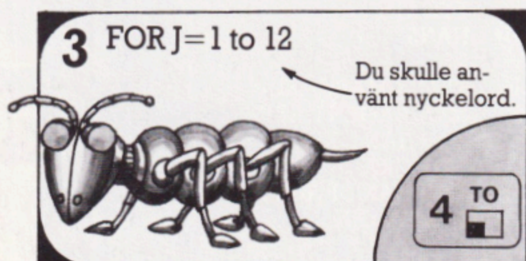
Hitta fel



Leta efter felstavningar i BASIC-ord. Datorn förstår inte felstavade ord.



Kontrollera 0 och O samt 1 och I för att se att du har skrivit rätt.



Om du har en Sinclairdator får du kontrollera att du inte skrev in ordet genom att skriva tecken för tecken istället för att trycka på tangenten för det ordet.

Skriva program

Kom ihåg när du skriver program att datorn kan utföra tre huvudaktiviteter: utföra enkla instruktioner, upprepa dessa och fatta beslut. Det är de viktigaste ingredienserna i alla program.

Enkla instruktioner

```
LET A=3
LET N=N+1
PRINT A/T
PLOT (X,Y)
```

Upprepning

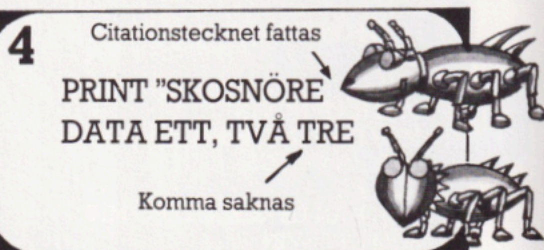
```
FOR J=1 TO 6

20 LET A=1
30 IF A<10 THEN
GOTO 100
```

Fatta beslut

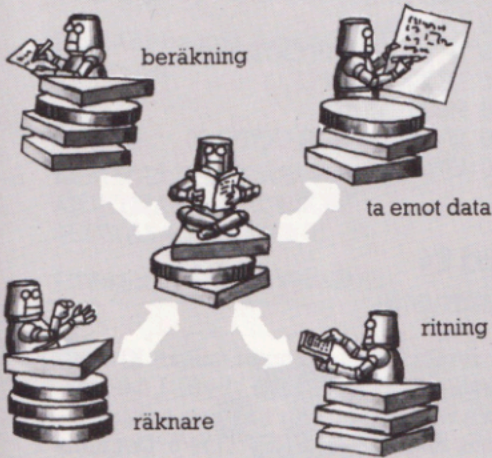
```
IF X=Y THEN STOP
IF K$ = "HEJ"
THEN PRINT A
```

I denna bok har vi visat dig alla viktiga instruktioner i BASIC som du behöver för att få datorn att utföra dessa aktiviteter. När du skriver program börja alltid med att planera vad datorn ska göra i varje del av programmet och bestäm sedan vilka instruktioner som skall användas.

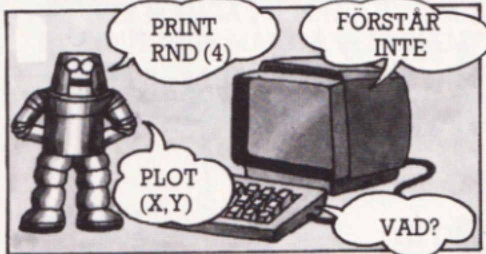


Kontrollera att du inte har glömt några citations- eller kommatecken. Kontrollera extra noga komplicerade programrader.

Det finns vanligen flera olika sätt att skriva program på. En del ger enklare och kortare program. När du skriver ett långt program kan det vara idé att dela det i flera delar med subrutiner som utför olika aktiviteter. Ryggraden i ett program kan vara en enkel uppsättning av instruktioner, beslut och upprepningar som bestämmer när och hur ofta datorn ska utföra subritinerna.



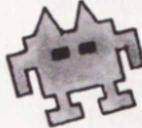
Om du delar in programmet i delar blir det mycket enklare att hitta fel. Varje del kan vanligen testas separat utan att hela programmet behöver köras. Kom ihåg att skriva en kommentar (REM) för varje del så att du kommer ihåg vad den gör.



Kontrollera att du givit de rätta RND, PLOT och CLS instruktionsorden för datorn. Kontrollera också att du givit datorn ett grafikprogram om den behöver det.

Felmeddelanden

Alla datorer skriver ut felmeddelande när de hittar fel i programmet. Dessa meddelanden finns förklarade i handboken. Här kommer några av de oftast använda meddelandena du kan få.



Detta betyder att det inte finns tillräckligt med data i DATA-raden. Det kan ha orsakats av att du har missat ett kommatecken i raden.



Raden med numret som finns i GOTO eller GOSUB satsen finns inte. Du kanske oavsiktligt raderade bort raden eller skrev fel nummer.



Ett sådant meddelande kan du få från en BBC- eller Sinclairdator. Det betyder att du har glömt initiera variabeln med en rad med LET C=0 eller LET C="" innan du försöker använda den.



Det här betyder att NEXT-raden saknas i loopen. Kanske för att du skrev in fel variabelnamn eller skrev 1 istället för I.

Sista ordet

Vissa fel är mycket svåra att upptäcka, men fungerar inte programmet då betyder det att det finns ett fel där. Kan du inte hitta felet får du försöka mata in de misstänkta raderna igen, det kanske blir rätt då.

Svar till övningsuppgifter

Sid 15
Namn-programmet

```
10 PRINT "VAD HETER DU?"
20 INPUT N$
30 PRINT "HEJ"
40 PRINT N$
50 PRINT "HUR MÅR DU?"
```

Sid 17
1. Beräkningsprogrammet

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+3
70 PRINT A*B,A/B
80 END
```

Komma för mellanrummet

2. Tabellprogrammet

Mellanrum

```
30 PRINT A;" GÅNGER ";B;" ÄR LIKA
   MED ";A*B
40 PRINT A;" DIVIDERAD MED ";B;" ÄR
   LIKA MED ";A/B
```

3. En variation till namnprogrammet

```
10 PRINT "VAD HETER DU?"
20 INPUT N$
30 PRINT " HEJ ";N$" HUR MÅR DU?"
```

Sid 18
Beräkningsprogrammet

```
10 PRINT "HUR MYCKET BLIR 7 GÅNG-
   ER 7"
20 INPUT A
30 IF A=49 THEN PRINT "KORREKT"
40 IF A<>49 THEN PRINT "NO ";7*7 ÄR
   LIKA MED 49"
```

Du måste ha semikolon här

Sid 19
Ålderprogrammet

Ersätt rad 30 och addera en ny rad 35

```
30 IF G<14 THEN PRINT "ÄLDRE ÄN
   SÅ"
35 IF G>14 THEN PRINT "YNGRE ÄN
   SÅ"
```

Sid 23
Ritprogrammet

```
5 LET C=0
45 LET C=C+1
50 IF C<6 THEN GOTO 10
```

Rita initialer

Här får du ett exempel hur du kan rita bokstaven L.

```
10 LET X=15
20 LET Y=30
30 PLOT (X,Y)
40 LET Y=Y-1
50 IF Y>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

Sid 24
Slumptal

Formeln för ett slumptal mellan 10 och 20 skulle bli INT(RND(1)*11+9). I datorer som endast behöver tal inom parentes efter RND blir det RND(11)+9. Det finns elva möjliga tal mellan 10 och 20, så du måste få fram ett slumptal mellan 1 och 11 och sedan addera 9.

Sid 25
Rymdattack

Dessa rader måste adderas till programmet för att hålla reda på antalet träffar.

```
15 LET S=0
75 IF X=A*B THEN LET S=S+1
95 PRINT "DU TRÄFFADE ";S;" AV 6
   FIENDER"
```

Sid 27
1. Gånger åtta-tabell

```
10 PRINT "GÅNGER ÅTTA-TABELL"
20 FOR J=1 TO 12
30 PRINT J;"×8=";J*8
40 NEXT J
```

Sid 27
2. Gånger N-tabell

```
10 INPUT "SKRIV IN ETT TAL";N
20 PRINT "HÄR KOMMER GÅNGER
   ";N;" -TABELLEN"
30 FOR I=1 TO 12
40 PRINT I;" GÅNGER ";N;" ÄR LIKA
   MED ";I*N
50 NEXT I
60 INPUT "ETT ANNAT TAL (Y ELLER
   N)";M$
70 IF M$="Y" THEN GOTO 10
```

För ZX81 behövs separata PRINT och INPUT rader.

Sid 32

LEFT$(A$,5) är "DATOR"
RIGHT$(A$,6) är "ORBOK"
MID$(A$,4,7) är "ORBO"

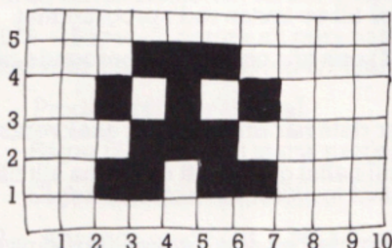
Sid 34
Trickprogrammet

```
10 PRINT "TÄNK UT ETT TAL"
20 PRINT "FÖRDUBBLA DET, ADDERA /
   4"
30 PRINT "DIVIDERA DET MED 2, AD-
   DERA 7"
40 PRINT "MULTIPLICERA MED 8,
   SUBTRAHERA 12"
50 PRINT "DIVIDERA MED 4 OCH TAG
   BORT 11"
60 PRINT "GE MIG RESULTATET"
70 INPUT N
80 PRINT "TALET DU FÖRST TÄNKTE
   PÅ VAR"; (N-4)/2
```

Du behöver parentesen för att ta siffrorna i den ordning som du vill.

Invasion från rymden

1



Rita på ett rutat papper upp en enkel figur som skall symbolisera en rymdvarelse.

2

4,5; 5,5; 6,5

3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Arbeta sedan fram koordinaterna för varje fyrkant som bildar rymdfiguren.

3

```
5 CLS
50 "HUR MÅNGA PUNKTER HORISONTELLT?";W
60 INPUT "HUR MÅNGA VERTIKALT?";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6

130 NEXT J
140 NEXT I
150 END
```

Ändrar 6 till ett högre tal för att få figuren att upprepas på skärmen flera gånger. (Har du fått ett fel blev det för högt tal.)

Skriv dina ritrader in här.
90 PLOT (J+3, I+2)
92 PLOT (J+4, I+2)
för punkten längst ner till vänster på figuren. Du måste ha en programrad för varje punkt.

Kopiera det här programmet utom rad 10 till 40 och 90 till 140 som ovan. Dessa rader ger ett slumpmönster, så du behöver dem inte. Nu får du skriva in dina egna rader med ritinstruktioner mellan raderna 80 till 140. (Du kan numrera om dessa rader i ditt program.) För varje koordinatpar måste du addera J och I för att få figuren att upprepas.

BASIC-ord

Här kommer en förteckning med korta förklaringar över BASIC-ord som används i den här boken. Vissa av dessa ord, som t ex CLS, är inte standard för alla datorer. Dessa ord har markerats med en asterisk. Om du har en dator får du kontrollera i handboken vilka ord som är aktuella för dig.

⋆ BREAK – På en del datorer stoppas programexekvering med detta ord. Men du måste vara försiktig, för på några datorer raderas hela programmet med det ordet. Där ska du använda ESCAPE eller något annat ord.

⋆ CLS – Gör rent på skärmen.

⋆ DATA – En förteckning över ord eller tal som skall lagras i form av variabler. Se även READ.

DIM – Talar om för datorn hur stort utrymme som skall reserveras för en variabel. T ex betyder DIMAX (5,4) att variabeln behöver 5 rader med fyra kolumner.

⋆ EDIT – Möjliggör en ändring i en programrad utan att hela raden behöver skrivas om.

⋆ END – Talar om för datorn att här slutar programmet. En del datorer behöver END-instruktion, andra, som t ex Sinclair, klarar sig utan den.

FOR...NEXT – Får datorn att gå runt i programmet och upprepa varje instruktion i loopen ett bestämt antal gånger.

GOSUB – Får datorn att lämna huvudprogrammet och gå ut till s k subrutin.

GOTO – Talar om för datorn att den ska gå till en annan rad i programmet.

IF...THEN – Jämför data (t ex tal eller ord) och gör olika saker beroende på resultatet.

INPUT – Ett sätt att få datorn att fråga efter data när programmet exekveras.

INT – Omvandlar tal med decimalpunkt till heltal genom att utelämna allt till höger om decimalpunkten. T ex INT (3.40)=3

⋆ LEFT$ – Säger till datorn att göra någonting med ett antal tecken från vänster i strängen.

LEFT$(A$,4) betyder tag bort fyra tecken från vänster i A$.

LEN – Talar om variabellängden, dvs antalet tecken i en variabel.

DATAORD

Bug – fel i programmet.
CPU – centralenheten i en dator. Där sker allt arbete med data och därifrån styrs datorns alla funktioner.
Flödesschema – visar de viktigaste delarna i ett program. Används ofta som ett hjälpmedel för att skapa program.
Fält – ett område i minnet innehållande variabler med data.
Grafik – ett sätt att visa information på skärmen (istället för text).
Kilobyte (K) – är en storhet som används för att tala om den tillgängliga minneskapaciteten. IK är lika med 1024 byte och en byte behövs på de flesta datorer för lagring av ett tecken.
Markör – (cursor) en ljuspunkt på skärmen. Den har olika utseenden på olika datorer. Används för att visa vart nästa tecken skall hamna på skärmen.
Prompt – ett fråge- eller annat tecken som visas på skärmen när datorn frågar om mer information efter ett INPUT-kommando.

LET – sätter variabelnamn på ett minnesutrymme och lagrar informationen där. T ex LET N=4 eller LET B$="KATTER"

★ LIST – Listar programmet på skärmen.

★ MID$ – Säger till datorn att göra någonting med tecknen i mitten av strängen. MID$(A$,4,3) betyder tag tre tecken från A$, börja med det fjärde.

NEW – Rensar programminnet för att göra plats för ett nytt program.

★ NEWLINE KEY – Talar om för datorn att du har avslutat inmatning av en programrad eller information. På en del datorer kallas samma tangent för RETURN eller ENTER.

NEXT se FOR.

★ PLOT – Säger till datorn att tända en punkt på skärmen. PLOT(X,Y) betyder tänd en punkt med koordinaterna X och Y.

PRINT – Får datorn att visa informationen på skärmen (eller skrivaren).

★ READ – Säger till datorn att läsa informationen från DATA-raden och lagra den i en variabel. Se DATA.

★ READY – En del datorer visar READY när de är klara att ta emot en ny instruktion.

REM – Datorn ignorerar rader som börjar med REM, men visar dessa i programlistningen. Används för att skriva kommentarer om vad olika delar av programmet gör.

RETURN – I slutet av en subrutin. Säger till datorn att gå tillbaka till huvudprogrammet, till instruktionen direkt efter den med GOSUB-satsen.

★ RIGHT$ – Säger till datorn att göra någonting med tecknen till höger i strängen. T ex betyder RIGHT$(A$,4) tag fyra sista tecknen i A$.

★ RND – Producerar ett slumptal.

RUN – Säger till datorn att starta programmet.

SQR – Säger till datorn att räkna ut kvadratroten.

STEP – Används med FOR...NEXT-loopar. Talar om när en loop skall upprepas.

STOP – Används för att stoppa programexekvering.

THEN – Se IF.

★ UNPLOT – Får datorn att släcka den tända punkten.

Pixel – betyder bildelement och är den lilla punkten som datorn kan tända och släcka på skärmen för att bygga upp bilder.
Program – en numrerad lista av instruktioner som talar om för datorna hur ett uppdrag skall utföras.
RAM – Random Access Memory, betyder arbetsminnet. Där kan både data och program lagras. Allt försvinner när strömmen slås av.
ROM – Read Only Memory, ett minne vars innehåll endast kan läsas av, men inte ändras.
Sträng – en serie av tecken som skall lagras i en variabel, t ex "KORV" eller "ABC 123".
Subrutine – en avgränsad del av programmet med egen funktion.En subrutin kan användas på mer än ett ställe i programmet.
Syntaxfel – ett fel (t ex stavfel) i BASIC-programmet, som innebär att du brutit mot någon regel på programspråket.
Variabel – ett namngivet minnesutrymme. Det innehåller data som kan ändras då programmet körs.

Gå vidare

Det bästa sättet att lära sig skriva program är att prova dem på en dator. Har du ingen dator kanske kan du gå någonstans där det finns en som du får använda. Fråga i skolan eller kontakta en av många datorklubbar. Kanske får du komma in och prova.

Du kan också lära mycket av att läsa andras program eller mata in dessa program i en dator.

Här får du en förteckning över böcker om datorer och programmering som du kanske kan ha nytta av.

Bli vän med hemdatorn Svensk text och redaktion: Tad Gruber och Gull-Mari Lenderud
Dator- och videospel Svensk text: Johan Wahlén. Granskning och och teknisk konsultation: Tad Gruber och Gull-Mari Lenderud
Rymdspel Svensk text: Johan Wahlén. Granskning och teknisk konsultation: Tad Gruber och Gull-Mari Lenderud
Stridsspel Svensk text: Johan Wahlén. Granskning och teknisk konsultation: Tad Gruber och Gull-Mari Lenderud

Index

Originalets titel: Introduction to Computer Programming
Engelsk text: Brian Reffin Smith och Lisa Watts
Svensk text och redaktion: Tad Gruber och Gull-Mari Lenderud
Illustrationer och layout: Graham Round och Martin Newton
Omslagslayout: Leon Zetherström

ISBN 91-7608-208-3
Seriens idé, layout, illustrationer:

Sättning:
Dahlberg & Co Tryckeri AB, Stockholm
Printed in Belgium

PRINTED IN BELGIUM BY proost INTERNATIONAL BOOK PRODUCTION

Hemdator-serien

Det är roligt med datorer. Du kan spela spel med dem, ställa frågor till dem, skriva poesi med dem och även spela musik på dem. Den här nya serien visar dig vilka spännande saker datorer kan göra. Här får du veta hur datorer arbetar och hur du kan använda dem. Böckerna är rikt illustrerade i färg och är skrivna på ett klart och enkelt språk som lätt kan förstås även av nybörjare.





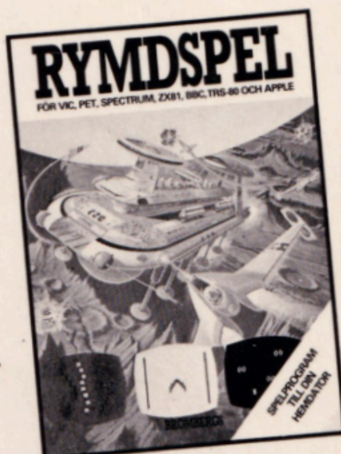


Bli vän med hemdatorn
En färgglad introduktion till hemdatorer: om hur de arbetar och vad de kan åstadkomma. Här hittar du massor av idéer om vad du kan göra med din hemdator.

Introduktion till programmering
En vägledning för nybörjare som vill lära sig att programmera med BASIC. Program som går att utföra på vilken hemdator som helst.

Datorspel och videospel
Här en titt på hur datorer spelar Space Invaders, schack och många andra spel. Med mängder av tips om hur du vinner förstås.

Rymdspel och Stridsspel
Dessa två utomordentligt fint illustrerade böcker är fullmatade med spelprogram för en hemdator. Varje spel kan användas på de vanligaste hemdatorerna. Många tips och fingervisningar för att göra egna program.





BROMBERGS
ISBN-91-7608-208-3